L'Assembler per il COMMODORE 64 e il VIC-20

K. Skier

L'Assembler per il COMMODORE 64 e il VIC-20

L'Assembler per il COMMODORE 64 e il VIC-20

K. Skier

McGRAW-HILL Book Company GmbH

Amburgo · New York · St Louis · San Francisco · Auckland · Bogotá · Città del Guatemala · Città del Messico · Johannesburg · Lisbona · Londra · Madrid · Montreal · Nuova Delhi · Panama · Parigi · San Juan · San Paolo · Singapore · Sydney · Tokyo · Toronto

Un libro di **BYTE**

Ogni cura è stata posta nella creazione, realizzazione, verifica e documentazione dei programmi contenuti in questo libro. Tuttavia né l'Autore né la McGraw-Hill Book Co. possono assumersi alcuna responsabilità derivante dall'implementazione dei programmi stessi, né possono fornire alcuna garanzia sulle prestazioni o sui risultati ottenibili dal loro uso, né possono essere ritenuti responsabili di danni o benefici risultanti dall'utilizzo dei programmi. Lo stesso dicasi per ogni persona o società coinvolta nella creazione, nella produzione e nella distribuzione di questo libro.

Titolo originale: *Top-Down Assembly Language Programming for Your VIC-20 and Commodore 64*

Realizzazione editoriale: EDIGEO srl, via del Lauro 3, 20121 Milano
Traduzione: Alessandra Carmignano e Lucio Oriani
Grafica di copertina: Valentina Boffa
Composizione e stampa: Litovelox, Trento

ISBN 88-7700-011-2

1ª edizione Marzo 1985

Indice

Introduzione

Generalmente si sente dire che i computer stanno diventando sempre più "intelligenti"; ma i computer, in quanto macchine, non sono per nulla intelligenti, lo sono i programmatori, cioè quelle persone che insegnano ai microprocessori a risolvere i più diversi problemi.
La stesura dei programmi è quindi di fondamentale importanza per ottenere il massimo dalla macchina. Tenendo presente queste considerazioni, questo libro si prefigge due scopi ben precisi: innanzitutto fornire ai principianti alcune tecniche, terminologie e conoscenze dell'Assembler in generale e del 6502 in particolare; in secondo luogo di presentare una serie di routine utili nello sviluppo di programmi in codice macchina per il 6502.
Il primo capitolo vuole essere una rapida guida attraverso l'hardware e il software del vostro computer. I capitoli 2, 3 e 4 costituiscono un rapido corso di Assembler destinato a quei lettori che si avvicinano all'argomento per la prima volta. Il resto del volume riporta i listati in codice sorgente, oggetto e Assembler di programmi che potrete inserire nel vostro computer.
I programmatori hanno a lungo cercato di sviluppare programmi compatti e veloci con lo sconsolante risultato che spesso il programma risulta illeggibile semplicemente perché il programmatore ha voluto risparmiare pochi byte o pochi cicli. In certi casi, quando la memoria è particolarmente ridotta o il tempo di esecuzione è critico, la leggibilità del programma viene sacrificata alle prestazioni.
Nella creazione dei programmi che compaiono in questo libro si è tenuto conto di numerosi obiettivi, molti dei quali più importanti della velocità di esecuzione o della dimensione di un blocco di codice. I programmi risulteranno quindi:

— **Utili:** nessun programma è presentato semplicemente per dimostrare una particolare tecnica di programmazione. Tutti i programmi del libro sono stati scritti per necessità effettive.
— **Facili da usare:** dando semplicemente un'occhiata allo schermo è possibile sapere quale programma si sta eseguendo e in quale modo. Quando l'esecuzione richiede un input dall'operatore, il programma specifica le informazioni necessarie e permette di correggere eventuali errori commessi nella risposta. Per usare questo software non dovrete ricordare indirizzi di programma o variabili. Le funzioni sono assegnate a tasti individuali ed è possibile stabilire i tasti funzione nella maniera che più vi aggrada.
— **Leggibili:** un principiante dovrebbe essere in grado di comprendere la logica di ogni singolo programma. Le label e i commenti nei listati sono stati scelti molto accuratamente per spiegare lo scopo di ogni variabile, subroutine e linea di programma.
— **Portabili:** il software qui presentato funziona indifferentemente sul C-64 e sul VIC-20. Con un adeguato adattamento è possibile usare questi programmi su qualunque computer basato sul 6502.
— **Compatibili:** fintanto che altro software non fa uso del quarto blocco di 4K della memoria (locazioni da 3000 a 3FFF) non dovrebbe esistere conflitto tra il vostro software e le routine di questo libro. In particolare, la maggior parte dei programmi presentati lascia inalterati i puntatori della pagina zero, cosicché non ci sarà la necessità di salvarli e ripristinarli prima e dopo ogni chiamata alla subroutine.
— **Espandibili:** i programmi di questo libro sono altamente modulari e possono essere estesi o ristrutturati per meglio soddisfare i vostri bisogni. Le subroutine specifiche del sistema vengono richiamate indirettamente, cosicché è possibile sostituirle con altre e la maggior parte dei valori può essere trattata come variabile. Non esistono programmi "monolitici" in questo libro, sono tutte subroutine che è possibile combinare in molte maniere per ottenere nuove potenti strutture.
— **Compatti:** sappiamo che ogni personal computer ha una certa disponibilità di memoria (troppo poca) e conosciamo anche il modo di scrivere programmi che occupino il 10 o il 20 per cento in meno di spazio, ma così facendo si sacrificherebbe la leggibilità, portabilità o espandibilità. In molti casi abbiamo temuto, per risparmiare un byte, di perdere la chiara comprensibilità della logica del programma e questo sarebbe stato un prezzo troppo alto da pagare. Attenzione: se usate un VIC-20 è necessaria l'espansione da 8K.
— **Veloci:** il software di questo libro è stato scritto per operare il più velocemente possibile tenendo presenti gli obiettivi sopra esposti. Ma ogni volta che è stato necessario operare una scelta tra chiarezza e velocità, abbiamo penalizzato quest'ultima. Un programma veloce ma illeggibile non è di alcun valore. Nessun programma in questo libro si

preoccupa di quanto dovrete attendere, ma cerca invece di essere il più chiaro possibile. Inoltre, come potreste stabilire se state aspettando un millisecondo in più?
Andate avanti, leggete, programmate, ma, soprattutto, divertitevi!

Nota

Questo testo ha una duplice valenza: da una parte è una completa ed esauriente descrizione del linguaggio Assembler e dall'altra una guida alla realizzazione di un assemblatore completo e funzionante.
Questo progetto, che si snoda lungo i vari capitoli, consente una comprensione, dall'interno, dei concetti e della programmazione in codice macchina e l'assemblatore che ne risulta li esemplifica nella pratica.
Ovviamente, questo programma non è confrontabile con un assemblatore/disassemblatore professionale che consente un uso più semplice ed immediato; consigliamo perciò agli utenti che vogliono programmare intensivamente in codice macchina, l'acquisto della cassetta *Assembler/Disassembler per il Commodore 64* di A. Bleasby, prodotta dalla *McGraw-Hill.*

Capitolo

La struttura del computer

1

Il C-64 e il VIC-20 sono macchine molto potenti, ma per poter sfruttare completamente questa potenza, dobbiamo innanzitutto capire come lavorano. Così, prima di iniziare a programmare daremo uno sguardo veloce al computer.

Il microprocessore 6502

Inizieremo con il 6502 che è quel componente del vostro sistema che svolge i calcoli. Preso singolarmente, il 6502 non può fare molto; dispone di tre registri (speciali aree di memoria che contengono i dati su cui il programma opera) chiamati A, X e Y e che possono contenere valori compresi tra 0 e 255. Questi registri hanno diverse caratteristiche, per esempio a un valore contenuto in A (accumulatore) è possibile addizionare o sottrarre qualsiasi valore compreso tra 0 e 255 mentre i valori contenuti in X e Y possono essere aumentati o diminuiti solo di una unità.
Il 6502 può anche trasferire il contenuto di un registro in un altro oppure in una qualsiasi locazione di memoria, come pure compiere il procedimento inverso, cioè assegnare ad un registro il valore contenuto in una locazione di memoria. Sfruttando queste capacità, il 6502 può operare su vari valori contemporaneamente, caricando i registri con i numeri contenuti nelle varie locazioni di memoria interessate, eseguendo le operazioni del caso e poi memorizzando il risultato in memoria.

Diversi tipi di memoria

Il computer memorizza i dati come una serie di 1 e 0 semplicemente perché la memoria non è altro che un'elaborata matrice di interruttori che possono avere solo due posizioni possibili, aperto (1) o chiuso (0). Non tutte le memorie però sono uguali: in quelle chiamate ROM (*Read Only Memory* - memoria a sola lettura) gli interruttori sono fissi, non possono cioè essere cambiati. In altre memorie, dette RAM (*Random Access Memory* - memoria ad accesso casuale), lo stato degli interruttori può essere cambiato dal microprocessore: il 6502 può infatti aprire o chiudere gli interruttori (bit) della memoria programmabile (RAM) e in seguito rileggere ciò che ha scritto. La Figura 1.1 mostra come la memoria viene utilizzata dal processore.

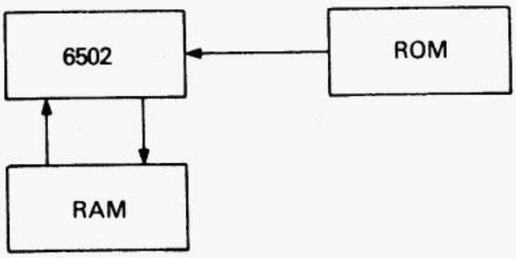


Figura 1.1 Interazione del processore con la memoria. Le frecce indicano il flusso dei dati

Esistono altre memorie dette *porte di input*, che vengono utilizzate dai dispositivi esterni come tastiere, terminali, ecc. Il 6502 elabora i dati che vengono immessi da questi dispositivi esterni leggendo le porte. Essendoci le porte di input, non potranno mancare quelle di output che permetteranno al computer di colloquiare con il mondo esterno.
Non pensate a questo punto che nel libro sia contenuta la ricetta per far parlare il vostro computer; infatti, troverete spesso modi di dire come: «il 6502 *vede*, *ricorda* o *sa cosa fare*»; naturalmente questo è impossibile, ma serve a rendere più chiare certe idee e più digeribili certi concetti.
Ritornando al discorso precedente, è normale raggruppare i due tipi di memoria descritti in un unico gruppo; infatti accadrà che alcuni dispositivi come registratore a cassette o floppy disk saranno connessi sia alle porte di input che a quelle di output; chiameremo dunque queste memorie con il nome di *porte di I/O* (input/output - ingresso/uscita).
La Figura 1.2 riassume gli accessi del microprocessore alle memorie e ai dispositivi periferici.
Anche il monitor collegato al vostro sistema può essere considerato come un particolare tipo di memoria, infatti il 6502 può leggere e scrivere sul-

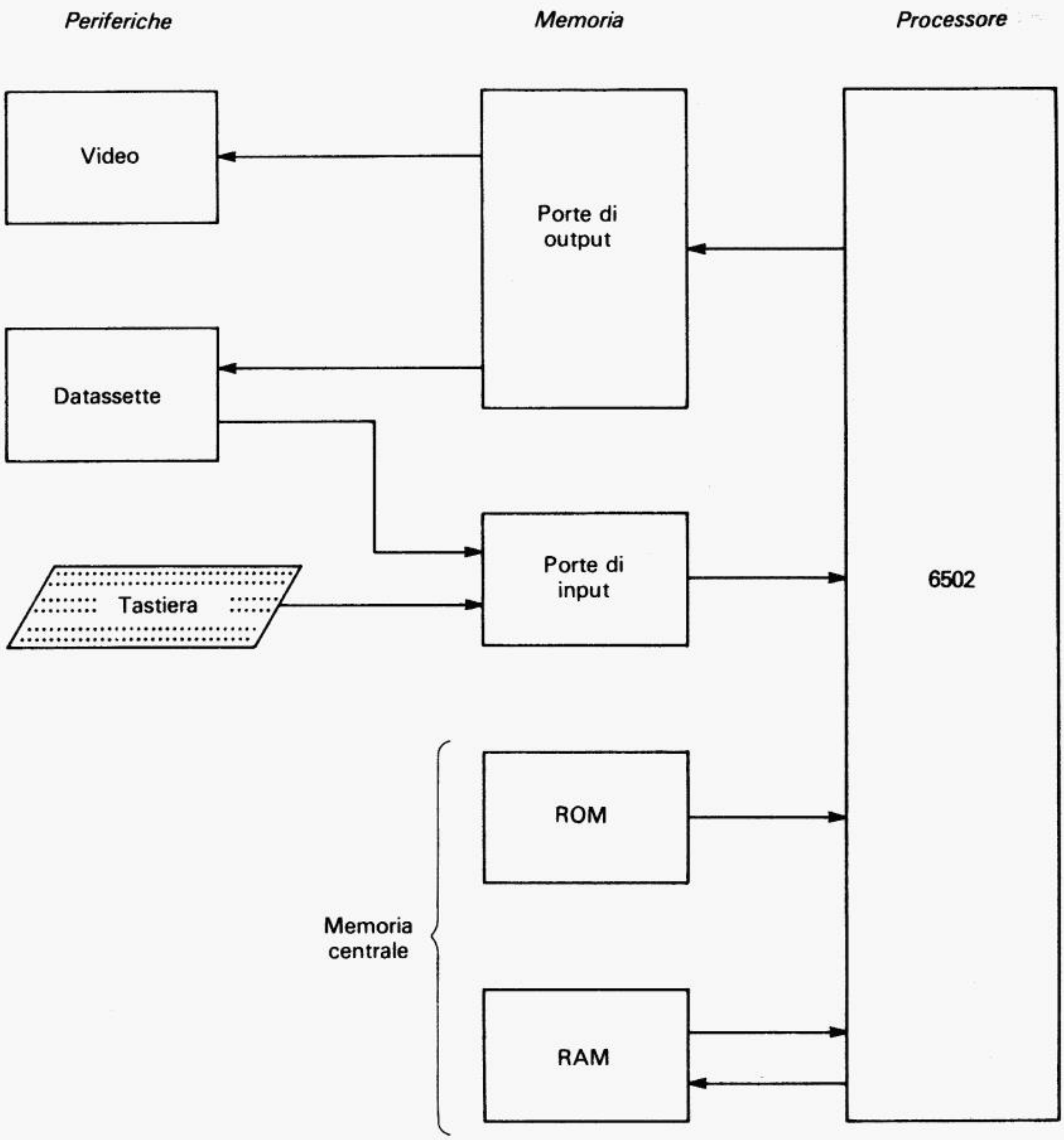


Figura 1.2 Accessi del 6502 ai dati in memoria e alle periferiche. Le frecce indicano il flusso dei dati

lo schermo. La tastiera viene scandita dalle porte di I/O che vengono decodificate, quindi la CPU può leggere la tastiera semplicemente testando alcune locazioni di memoria. Le porte di I/O sono di basilare importanza in ogni sistema perché permettono al microprocessore di colloquiare con l'utente come si può vedere in Figura 1.3.

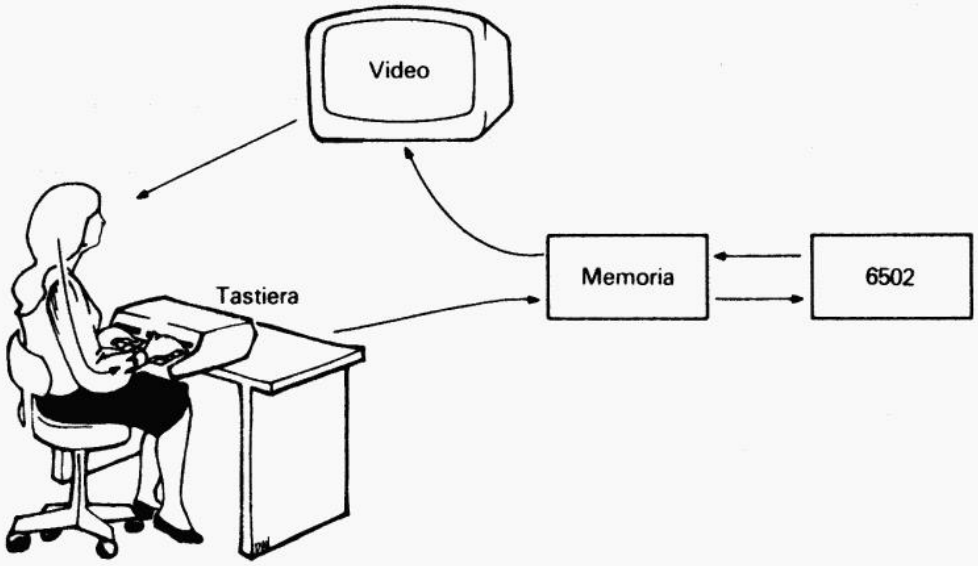


Figura 1.3 Come avviene la comunicazione tra 6502 e dispositivi periferici. Le frecce indicano il flusso dei dati

Il sistema operativo

Abbiamo per il momento parlato solo dell'hardware del computer, ma il C-64 e il VIC-20 non sono composti solo da questo.
Nel computer è memorizzato permanentemente nella ROM il sistema operativo che, tra le altre cose, include le routine di gestione dell'I/O necessarie per l'uso del monitor TV e della tastiera. Non ci interessa, in questa sede, il principio del funzionamento di queste routine, ma è necessario renderci conto che dobbiamo dipendere da esse per lo svolgimento di parecchie funzioni basilari.
Nel sistema operativo del vostro computer sono comprese molte altre routine che sarebbe utile usare direttamente, ma poiché il software presentato in questo libro è stato concepito per funzionare su diversi tipi di computer, si è cercato di sfruttare il sistema operativo al minimo indispensabile. In altre parole, i programmi contenuti nel libro non traggono alcun vantaggio dal sistema operativo della macchina, ma il software che voi svilupperete per vostro conto e che quindi sarà destinato a funzionare sulla vostra macchina dovrebbe avvalersi pienamente delle routine contenute nelle ROM del sistema operativo.

Il BASIC

Una delle caratteristiche principali del vostro computer è l'interprete BASIC contenuto nella ROM. Questo interprete non è altro che un programma che permette al computer di "capire" i comandi dati in linguaggio BASIC.
La documentazione allegata al vostro sistema vi fornisce una lista di comandi usabili in BASIC. Questo linguaggio, molto facile da imparare, vi permetterà di realizzare molti programmi, ma ogni istruzione BASIC dovrà essere analizzata e tradotta dall'interprete prima che la CPU possa eseguirla. La conseguenza di ciò è che un programma scritto in linguaggio BASIC non sarà mai molto veloce, o almeno non sarà mai così veloce come un programma scritto in codice macchina.

Il codice del 6502

La CPU (unità centrale di elaborazione) è il cuore del vostro computer. Il C-64 e il VIC-20 usano il 6502. Ogni microprocessore ha un suo gruppo di istruzioni ben definito. Questo insieme di istruzioni è a un livello molto inferiore dei comandi BASIC. Per esempio, in BASIC, possiamo scrivere una singola linea di programma come PRINT "CIAO" mentre in codice macchina ci vogliono parecchie istruzioni per ottenere lo stesso risultato. Una determinata sequenza di istruzioni in codice macchina fornirà lo stesso risultato su macchine diverse ma basate sullo stesso microprocessore. Così, se scrivete una sequenza di istruzioni del 6502 che compie una determinata funzione, questa verrà eseguita da qualunque calcolatore basato sul 6502, mentre non darà nessun risultato in macchine basate sull'8080 o sullo Z80 o sul 6800.
I programmi contenuti in questo volume sono tutti scritti nel codice del 6502; potranno quindi essere usati anche su Apple, PET, Atari e su tutte le macchine basate su questo microprocessore; inoltre richiedono mediamente 4K di memoria lasciandone quindi più che a sufficienza per i vostri programmi.

Capitolo

2 Introduzione all'Assembler

Avete mai provato a guardare un giocoliere? Le palline, le mazze o qualsiasi altra cosa egli lancia in aria si intrecciano in traiettorie così complicate che difficilmente è possibile distinguerle. Questo è molto bello, ma non è magia! È solo il risultato di parecchie ore di allenamento e dell'applicazione di determinate semplici regole. Con un buon allenamento e conoscendo le regole del gioco, anche voi potreste essere in grado di dare spettacolo.
La programmazione in linguaggio Assembler non è molto differente dal gioco dell'esempio; sarete infatti in grado di realizzare programmi che stupiranno voi stessi.
In parole povere le regole di un giocoliere sono quelle di afferrare un oggetto con una mano, lanciarlo in una determinata direzione e riprenderlo. Queste semplici regole sono applicabili anche alla CPU, il cui compito non è altro che prendere i dati nelle varie locazioni di memoria e spostarli in altre locazioni. In primo luogo una CPU è molto veloce e può eseguire parecchie istruzioni in poco tempo; inoltre, per mantenere il paragone con il giocoliere, quest'ultimo può contare solo su due mani, mentre il 6502 su tre (i registri A, X e Y) e su parecchie tasche (8000 o più byte di memoria).
Un byte è composto da 8 bit di dati che possono essere caricati contemporaneamente in un registro. Diremo quindi che un registro può contenere un byte come pure ogni locazione di memoria può contenere un byte. Il 6502 può operare su un singolo byte alla volta, ma poiché è in grado di eseguire centinaia di migliaia di operazioni al secondo, è anche in grado di operare su centinaia di migliaia di byte al secondo.

Rappresentazione binaria

Ogni valore viene memorizzato dal computer come una serie di bit. Se volessimo, potremmo rappresentare un byte per mezzo della sua configurazione; questo tipo di rappresentazione che fa uso solo di 1 o 0 viene chiamata *binaria*. Per esempio, il numero 25 decimale viene rappresentato in binario come 00011001.
Nel sistema binario ogni bit definisce la presenza o l'assenza di un determinato valore, inoltre ogni bit ha valore o "peso" doppio di quello immediatamente alla sua destra. Da questa affermazione deduciamo che il bit più a destra è il meno significativo (LSB - *least significant bit*) mentre quello più a sinistra è il più significativo (MSB - *most significant bit*).
La Tabella 2.1 chiarirà meglio il concetto e il peso di ogni bit componente un byte:

Tabella 2.1 Peso dei bit in un byte

Bit numero:	7	6	5	4	3	2	1	0
Peso:	128	64	32	16	8	4	2	1

Il bit più a destra (bit 0) ci dice se abbiamo un 1 nel byte, quello immediatamente alla sua sinistra (bit 1) ci dice se abbiamo un 2 e così via fino al bit più a sinistra (bit 7) che ci dice se nel nostro byte è contenuto un 128.
Il procedimento per trovare la rappresentazione binaria di un numero decimale, è molto semplice. Prendiamo ad esempio il numero 25; per prima cosa dobbiamo trovare la potenza del numero due che più si avvicina per difetto al valore, nel nostro caso è 2^4, cioè 16. A questo punto abbiamo $25-16=9$; tenendo nota che il bit 4 sarà un uno cerchiamo la potenza di due che più si avvicina per difetto al valore ottenuto, il risultato sarà 2^3, cioè 8. Anche il bit 3 sarà perciò un uno e il nostro valore sarà diventato $9-8=1$; quindi ci manca una unità per raggiungere il valore di partenza. La potenza del due che ha come valore 1 è 2^0, quindi anche il bit 0 sarà un uno. A questo punto il nostro valore di partenza 25 avrà rappresentazione binaria 00011001, infatti sommando i pesi dei bit a uno otteniamo $16+8+1=25$.
Il numero 25 può essere però espresso anche in altra maniera. Piuttosto che rappresentare un valore come una serie di 0 e 1 ci sarà più comodo usare la rappresentazione esadecimale.

Rappresentazione esadecimale

Contrariamente alla rappresentazione binaria che richiede un gruppo di otto caratteri per definire un valore a 8 bit, la rappresentazione esadecimale ne richiede solamente due. Questi caratteri non sono limitati a 0 e 1 ma includono tutta la serie dei numeri da 0 a 9 e le lettere dalla A alla F. In questo sistema abbiamo così un gruppo di 16 caratteri per rappresentare i valori.
Per rappresentare un byte di 8 bit in notazione esadecimale dobbiamo dividerlo in due byte da quattro bit detti *nybble*. Questi due byte da 4 bit hanno un valore compreso tra 0 e 15 (decimale) che possiamo esprimere con un singolo numero esadecimale. Il numero 10 decimale viene rappresentato in esadecimale con $A (il segno "$" indica per convenzione che il numero è espresso in notazione esadecimale). La Tabella 2.2 ci dà l'equivalenza decimale/esadecimale per i valori compresi tra 0 e 15.

Tabella 2.2 Equivalenza decimale/esadecimale

Carattere esadecimale	Decimale equivalente
$0	0
$1	1
$2	2
$3	3
$4	4
$5	5
$6	6
$7	7
$8	8
$9	9
$A	10
$B	11
$C	12
$D	13
$E	14
$F	15

Nell'Appendice G è contenuta la tabella delle equivalenze decimali/esadecimali per valori compresi tra 0 e 255.
In questo libro il codice oggetto di ogni programma sarà generalmente presentato in notazione esadecimale, quindi una chiara comprensione dei concetti della numerazione esadecimale è indispensabile per interpretare le istruzioni e per seguire lo svolgersi dei programmi.

Il codice ASCII dei caratteri

Un byte di 8 bit può essere usato anche per rappresentare una lettera dell'alfabeto maiuscola o minuscola, un segno di interpunzione o un carattere di controllo per la stampante, per esempio un ritorno a capo. Una stringa di questi caratteri può formare una parola, un messaggio o perfino un documento completo. Nell'Appendice A1 è contenuta la tabella dei caratteri ASCII e dei valori esadecimali corrispondenti. ASCII è l'acronimo di *American Standard Code for Information Interchange* (codice standard americano per lo scambio delle informazioni). Se, per esempio, volete memorizzare la lettera A in una qualsiasi locazione di memoria, guardando la tabella dell'Appendice A1, saprete che è necessario memorizzare il valore $41.
L'interpretazione di un byte come numero, carattere ASCII o altro, dipende esclusivamente dal programma che usa quel particolare valore.

Il ciclo macchina

Un microprocessore come il 6502 non può fare nulla in mancanza di precise istruzioni. Esso conosce solo 151 istruzioni, chiamate *opcode* (codici operativi). Ogni opcode è lungo un byte e può ordinare al microprocessore di svolgere una ben precisa operazione come, per esempio, memorizzare il valore di un registro in una locazione di memoria, oppure caricare

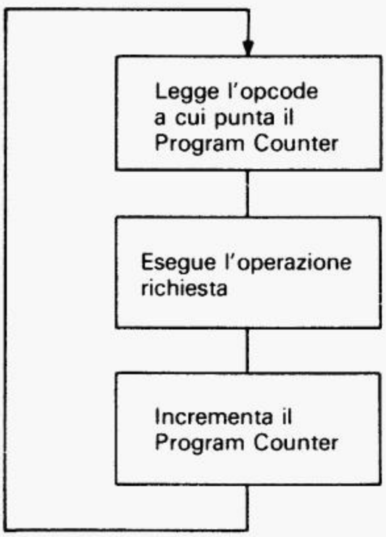


Figura 2.1 Il ciclo macchina del 6502

un registro con un valore prelevato dalla memoria, oppure compiere qualche altra operazione altrettanto semplice. L'Appendice A3 contiene una lista di tutti gli opcode del microprocessore 6502.
Cosa avviene durante lo svolgersi di un programma? Il 6502 legge un opcode, compie l'operazione specificata, legge il successivo opcode e svolge l'operazione specificata ..... e così via. Ma come può il microprocessore sapere dove si trova la successiva istruzione? Esiste un quarto registro, chiamato PC (*Program Counter* - contatore di programma) che contiene l'indirizzo di alcune locazioni di memoria. Quando il 6502 inizia un ciclo, legge l'istruzione dalla locazione di memoria specificata dal PC. Finito il ciclo, il PC punterà ad una nuova locazione di memoria che contiene l'istruzione successiva. La Figura 2.1 mostra il diagramma di flusso di un ciclo del microprocessore.

Il codice macchina

Un programma in codice macchina non è altro che una sequenza di istruzioni immagazzinate nella memoria del computer. Se è possibile fare in modo che il PC contenga l'indirizzo di partenza del nostro programma, allora diremo che il PC punta al nostro programma.
Quando il 6502 inizia il ciclo macchina, legge l'istruzione contenuta nella locazione di memoria puntata dal PC ed esegue il lavoro specificato. A questo punto potremo dire che il programma sta girando.
Ogni istruzione è memorizzata come un opcode di 1 byte che può essere seguito da uno o due byte di operando. Ad esempio, un programma in codice macchina per il 6502 potrebbe essere "A9 05 20 02 04 A2 F5 60".
A prima vista sembrerebbe solo una manciata di numeri (esadecimali naturalmente) ma è proprio questo codice che costituisce il "linguaggio" che la macchina può interpretare e da qui il nome di *codice macchina*.

Assemblatori

Se voi foste macchine vi sarebbe molto facile leggere i programmi direttamente in codice macchina; se la cosa vi risulta facile, forse appartenete a una nuova generazione di mutanti! Siccome non siamo macchine ma umani, sono stati sviluppati degli strumenti di programmazione, chiamati assemblatori, che accettano in ingresso un codice detto *sorgente* (più leggibile) e lo trasformano in un listato e in un codice macchina detto *oggetto*.
Un listato è naturalmente destinato a un essere umano poiché molto più

facilmente leggibile, mentre l'oggetto è una serie di istruzioni destinate al microprocessore.
A ogni programma contenuto in questo libro corrisponde una appendice che ne riporta il listato Assembler ed esadecimale. Il listato Assembler include il sorgente e l'oggetto, mentre il listato esadecimale vi mostra quello che viene letto dalla CPU.
La Figura 2.2 mostra come viene usato un assemblatore per produrre il listato per il programmatore e il codice oggetto per il computer.

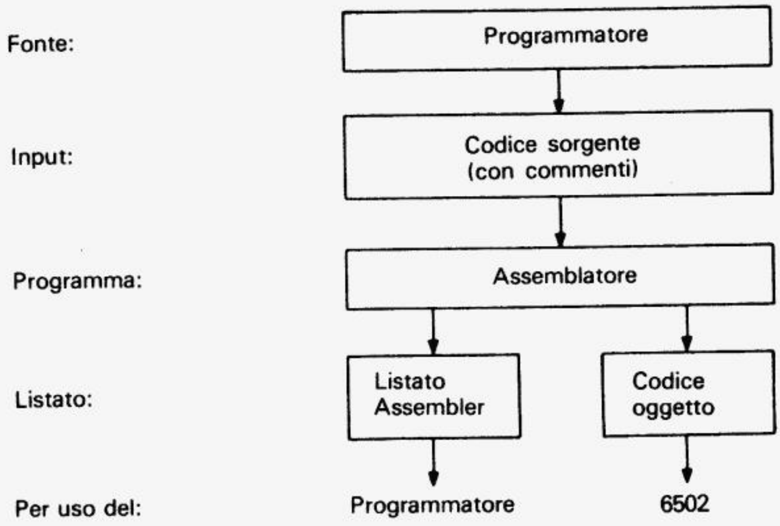


Figura 2.2 Dal programmatore al codice oggetto. L'assemblatore accetta il codice sorgente in ingresso e produce in uscita il listato Assembler e il codice oggetto

Il programma in formato sorgente

Un programma di questo tipo è costituito da una o più linee di codice sorgente. Una linea di codice sorgente è costituita da quattro campi:

LABEL MNEMONICO OPERANDO COMMENTO

Lo mnemonico, necessario in ogni caso, consiste di un gruppo di tre lettere scelte per ricordare la specifica funzione dell'istruzione rappresentata (naturalmente gli mnemonici sono in inglese). Per esempio, lo mnemonico LDA significa *LoaD Accumulator* (carica l'accumulatore); LDX significa *LoaD X* (carica il registro X); TXA significa *Transfer X register to the Accumulator* (trasferisce il registro X nell'accumulatore).

Questi codici mnemonici non sono così significativi come le istruzioni BASIC ma rappresentano un grande passo avanti rispetto agli opcode (vedi Appendice A2 per la lista degli mnemonici).
Alcune operazioni richiedono un operando: per esempio LDA richiede l'operando perché nella linea di programma bisogna specificare il valore con cui si vuole caricare l'accumulatore.
I campi label e commento sono opzionali. Una label permette di assegnare un nome a determinate locazioni di memoria, mentre i commenti non vengono inclusi nel programma oggetto ma hanno la stessa funzione delle REM in un programma BASIC, cioè quella di rendere più facile l'interpretazione del programma.
Quando scrivete un programma, anche se verrà letto esclusivamente da voi stessi, cercate di scegliere le label e i commenti in modo che chiunque possa capire la logica del programma. Questa semplice regola vi farà risparmiare molto tempo quando, a distanza di giorni, settimane o mesi, potreste non ricordare la funzione di certe linee prive di commento.

Caricamento di un registro

Proviamo a scrivere un programma semplicissimo per caricare in un registro un valore, per esempio carichiamo in A il numero 10. Poiché vogliamo caricare l'accumulatore, faremo uso dell'istruzione LDA (se il registro fosse stato X avremmo naturalmente usato LDX e così via). Sappiamo qual è lo mnemonico da usare per scrivere la nostra prima riga di programma, ma uno sguardo all'Appendice A5, che contiene gli opcode in ordine alfabetico e i modi di indirizzamento, ci dice che LDA ha diverse possibilità di indirizzamento. Che operando dovremmo usare nella nostra linea di programma?
Il nostro scopo è quello di caricare l'accumulatore con il numero 10, così dovremo usare l'indirizzamento immediato per caricare il numero 10 direttamente nell'accumulatore. Per indicare il modo immediato useremo il carattere "#":

Esempio 1

```
LDA #10
```

L'esempio 1 è una linea di programma sorgente che contiene solo 2 campi: uno mnemonico e un operando. Lo mnemonico, LDA, significa "carica l'accumulatore", ma con cosa? L'operando ci dice che cosa dovrà essere caricato nell'accumulatore. Il segno "#" specifica che l'operazione deve

essere eseguita in modo immediato il che significa che intendiamo caricare l'accumulatore con una costante specificata in questa linea di codice sorgente, piuttosto che con valori o variabili che si trovano in qualche locazione di memoria. Perciò l'operando identifica la costante da caricare nell'accumulatore, nel nostro caso 10.

Le costanti

Una *costante* è un valore conosciuto ed è immutabile nell'ambito del programma. Se un valore cambia durante l'esecuzione di un programma, viene allora detto variabile, ed è necessario riservare una o più locazioni di memoria per contenere il valore di ogni variabile.
Ci sono diversi tipi di costanti. Tutti i numeri sono delle costanti. Il numero 7, per esempio, è una costante: varrà sempre 7, oggi come domani. Un carattere è un altro tipo di costante, una "A" sarà sempre una A. Ma un gruppo di caratteri, come per esempio "Carburante" cambierà il valore rappresentato durante lo svolgersi del programma (per esempio la simulazione di un allunaggio), e così non è una costante.
Notate nell'esempio 1 che il segno "#" è l'unica interpunzione dell'operando. In assenza di speciali interpunzioni (come il segno "$" per indicare un numero esadecimale e il segno "'" per indicare un carattere ASCII) tutti i numeri verranno considerati decimali.
Che codice oggetto apparirà assemblando questa linea di programma? Proviamo ad assemblarla manualmente guardando l'Appendice A5 dove troveremo che l'opcode corrispondente a LDA/immediato è $A9. Il secondo byte deve specificare il valore da caricare nell'accumulatore. Vogliamo caricare il registro A con il valore decimale 10, che corrisponde a $0A; quindi il programma oggetto per l'esempio 1 è: A9 0A.
Quando questi 2 byte di codice oggetto verranno eseguiti dal 6502, l'accumulatore conterrà il valore $0A, che è in effetti quello che ci prefiggevamo. E se avessimo voluto caricare l'accumulatore con la lettera "M" piuttosto che con un numero? Avremmo lo stesso usato l'istruzione LDA e anche il modo immediato, specificando nell'operando la costante da caricare nell'accumulatore. Entrambe le linee seguenti otterranno lo scopo prefissato:

Esempio 2

```
LDA #'M
    oppure
LDA #$4D
```

In queste due linee di programma lo mnemonico e il segno # ci dicono che stiamo caricando l'accumulatore in modo immediato, cioè con una costante. L'operando che segue il # specifica la costante. Un apostrofo indica che seguirà un carattere ASCII, mentre un $ segnala un valore esadecimale. Nell'Appendice A1 troviamo che il carattere ASCI "M" equivale a $4D; sono semplicemente due maniere di rappresentare la stessa sequenza di bit. In questo modo le due linee di programma sono equivalenti; produrranno infatti lo stesso codice oggetto: A9 4D.
Quale di queste due linee sarà più leggibile? Se una costante deve essere usata in un programma come un carattere ASCII, sarà sicuramente meglio scriverla come un carattere ASCII.

Memorizzazione di un registro

Proviamo adesso a memorizzare il valore contenuto nell'accumulatore in una qualsiasi locazione di memoria. Ogni locazione di memoria ha un indirizzo unico (proprio come le case), che varia da $0000 a $FFFF. Supponiamo di voler memorizzare il contenuto dell'accumulatore all'indirizzo $020C. Per fare questo dobbiamo usare la seguente linea di programma sorgente:

Esempio 3

```
STA $020C
```

L'esempio verrà assemblato in questi 3 byte di codice macchina: 8D 0C 02. L'Appendice A5 ci dice che l'opcode per STA in modo assoluto è $8D. Quando il 6502 incontra 8D sa che deve memorizzare il contenuto dell'accumulatore nell'indirizzo specificato dai 2 byte che seguono. Il modo assoluto è usato quando si specifica un preciso indirizzo di memoria.
Nell'esempio l'indirizzo potrebbe sembrare sbagliato. Sembra che l'operando specifichi l'indirizzo $0C02, perché i byte sono in questo ordine: 0C seguito da 02. Ma noi vogliamo operare sull'indirizzo $020C. C'è forse qualcosa di sbagliato?

Prima il byte basso

È possibile pensare che ci sia qualcosa di sbagliato nello scrivere l'indirizzo $020C come 0C seguito da 02, ma evidentemente il 6502 "pensa" in modo differente.

Quando scriviamo un numero siamo abituati a scrivere la cifra più significativa per prima e la meno significativa per ultima, ma il 6502 non lavora in questo modo. Quando la CPU incontra due byte che rappresentano un indirizzo di memoria (*byte basso*) interpreta il primo come parte meno significativa dell'indirizzo e il secondo come parte più significativa (*byte alto*). Tutti i modi di indirizzamento che richiedono un operando di 2 byte, li vogliono in quest'ordine: prima la parte meno significativa e poi la parte più significativa. È da notare però che non tutti gli indirizzamenti richiedono un operando di due byte.

Indirizzamento in pagina zero

La memoria contenuta nel calcolatore è suddivisa in pagine, dove una pagina rappresenta un blocco di 256 byte consecutivi. La pagina compresa tra le locazioni $0000 e $00FF viene chiamata *pagina zero* perché tutti gli indirizzi di questa pagina hanno il byte alto uguale a $00. L'indirizzamento in pagina zero trae profitto proprio da questo fatto. Il codice sorgente che fa uso dell'indirizzamento in pagina zero una volta assemblato richiederà un solo byte nell'operando proprio perché l'opcode specifica l'indirizzamento in pagina zero e il byte alto viene sottinteso essendo sempre e comunque $00. Un indirizzo della pagina zero potrà quindi essere specificato in modo assoluto o in modo pagina zero tenendo però presente che in quest'ultimo caso l'operando sarà di un solo byte.
Se volete usare una locazione di pagina zero, per esempio $00F4, potreste scrivere:

Esempio 4

```
STA $00F4
    oppure
STA $F4
```

che assemblati darebbero, il primo: 8D F4 00 e il secondo: 85 F4.
L'opcode 85 significa: memorizza l'accumulatore in pagina zero alla locazione assoluta indicata ($00F4).

Espressioni simboliche

Mettiamo il caso di voler trasferire valori contenuti nelle locazioni $0200, $0201 e $0202 rispettivamente nelle locazioni $0300, $0301 e $0302. Potremmo scrivere queste linee di programma sorgente:

Esempio 5

```
LDA $0200
STA $0300
LDA $0201
STA $0301
LDA $0202
STA $0302
```

Questo programma carica alternativamente un byte nell'accumulatore e lo scarica in un'altra locazione di memoria. Da notare il fatto che caricando un registro questo viene cambiato mentre non viene cambiato il contenuto della locazione di memoria da cui si preleva il valore.
In alternativa avremmo potuto scrivere il seguente programma che fa riferimento agli indirizzi come espressioni simboliche:

Esempio 6

```
1    ORIGINE          = $0200
2    DESTINAZIONE     = $0300
3    LDA              ORIGINE
4    STA              DESTINAZIONE
5    LDA              ORIGINE+1
6    STA              DESTINAZIONE+1
7    LDA              ORIGINE+2
8    STA              DESTINAZIONE+2
```

Nell'esempio 6 le linee 1 e 2 sono direttive per l'assemblatore che assegnano alle locazioni $0200 e $0300 le label rispettivamente di ORIGINE e DESTINAZIONE.
In questo modo le linee che seguono possono fare riferimento a questi indirizzi per mezzo delle label o per mezzo di espressioni simboliche costituite dalle label e opzionalmente da costanti e operatori aritmetici. Il programma sorgente dell'esempio 6, una volta assemblato, produrrà lo stesso risultato del programma sorgente dell'esempio 5; ma l'esempio 6,

che fa uso di espressioni simboliche, risulterà molto più leggibile dell'esempio 5 dove gli operandi vengono specificati in esadecimale.

Alcuni esercizi

1. Scrivete le istruzioni necessarie per caricare l'accumulatore con il valore decimale 127, per caricare il registro X con la lettera "r" e per caricare il registro Y con il contenuto della locazione $B092.
2. Scrivete le istruzioni necessarie per trasferire il contenuto della locazione $0043 nella locazione $0092.

Capitolo

Loop e subroutine 3

Indirizzamento indicizzato

Anche se funzionante, il programma dell'esempio 6 non è molto efficiente poiché richiede due linee di codice sorgente per muovere ogni byte. Se dovessimo per esempio muovere 50 o 100 byte ci occorrerebbero 100 o 200 linee di programma, il che non è molto conveniente.
Fortunatamente esiste un modo di indirizzamento, chiamato indicizzato, che fa al caso nostro. In questo modo, piuttosto che specificare l'indirizzo assoluto o di pagina zero su cui operare, possiamo stabilire un *indirizzo di base* e un *registro indice*. Il 6502 sommerà il valore contenuto nel registro indice all'indirizzo di base, ottenendo come risultato la locazione dove è richiesta l'operazione. Perciò, se dovessimo spostare 9 byte da una locazione ad un'altra, potremmo farlo nel modo seguente, sfruttando il registro X come indice:

Esempio 7

```
        ORIGINE=$0200
        DESTINAZIONE=$0300

INIZIO  LDX #0              Pone X=0 per partire con il
                            primo byte
LEGGE   LDA ORIGINE,X       Legge l'X-esimo byte dal
                            blocco di origine
```

SCRIVE	STA DESTINAZIONE,X	Lo scrive nell'X-esima posizione del blocco di arrivo
SOMMA	INX	Incrementa X di un'unità per il byte successivo
TEST	CPX #9	Controlla se sono stati mossi 9 byte
SALTA	BNE LEGGE	Se X è diverso da 9 ritorna e legge il successivo

Useremo l'esempio 7 per spiegare alcune nuove istruzioni e modi di indirizzamento. L'esempio 7 è composto da sei linee di codice sorgente che permettono di spostare il contenuto di nove locazioni di memoria consecutive.
Se avessimo cercato di muovere questi nove byte con la tecnica illustrata negli esempi 5 e 6 sarebbero state necessarie 18 linee di programma.
Per mezzo dell'indirizzamento indicizzato, invece, abbiamo risparmiato ben 12 linee di programma; cerchiamo ora di capire come funziona questo tipo di indirizzamento. Tutte le linee sono etichettate, quindi possiamo analizzarle una ad una.
L'istruzione con label INIZIO carica il registro X in modo immediato con il valore 0. Dopo aver eseguito questa operazione sappiamo per certo che X=0 ma non sappiamo nulla circa gli altri registri.
La linea LEGGE carica l'accumulatore con il valore contenuto nell'X-esima locazione di memoria a partire dall'ORIGINE, quindi, al primo passaggio da questa linea il registro X conterrà 0 e l'accumulatore verrà caricato con il valore contenuto nella locazione (ORIGINE+0).
In ogni linea di programma sorgente, una virgola nell'operando indica che si farà uso dell'indirizzamento indicizzato; quindi, una virgola seguita da X significa che il registro X è l'indice dell'istruzione.
Nell'indirizzamento indicizzato ci sono diversi modi. Due di questi sono *indirizzamento indicizzato assoluto* e *indicizzato in pagina zero.*
La linea LEGGE dell'esempio 7 usa l'indirizzamento indicizzato assoluto se l'ORIGINE si trova sopra la pagina zero, ma se l'ORIGINE fosse in pagina zero si può adoperare l'indirizzamento indicizzato in pagina zero che come l'indirizzamento in pagina zero richiede un solo byte nell'operando.
Nell'indirizzamento indicizzato, l'operando contiene l'indirizzo di base e il 6502 opererà sulla locazione che si ottiene sommando l'indirizzo di base al valore contenuto nell'indice (X o Y). Il solo caso in cui il microprocessore opera sull'indirizzo di base è quando il registro indice è uguale a zero; in caso contrario l'indirizzo sarà una locazione di memoria più alta.
A questo punto del programma ci troviamo con l'accumulatore che contiene il valore della locazione ORIGINE. Nella linea SCRIVE, l'accumulatore verrà scaricato nell'X-esima locazione di memoria a partire dalla

DESTINAZIONE. Poiché non abbiamo fatto nulla per cambiare il valore dell'indice X, questo sarà ancora uguale a 0 e quindi l'accumulatore verrà scaricato proprio nella locazione DESTINAZIONE.
Abbiamo così trasferito il valore dalla locazione ORIGINE alla locazione DESTINAZIONE e l'indice è ancora uguale a 0.
Nella linea SOMMA, che per il momento è la più corta incontrata, troviamo l'istruzione INX che significa INcrementa X; siccome fino a questo punto X era uguale a 0, adesso sarà uguale a 1.

L'istruzione CMP

Nell'esempio 7 la linea TEST compara il valore contenuto nel registro X con il numero 9. Ci sono 3 istruzioni di comparazione per il 6502, una per ogni registro; CMP compara un valore con il contenuto dell'accumulatore, CPX compara un valore con il contenuto del registro X e CPY compara un valore con il contenuto del registro Y.
Possiamo usare queste istruzioni per comparare qualsiasi registro con qualsiasi valore in memoria o, in modo immediato, per comparare un qualsiasi registro con una costante.
Queste comparazioni ci permettono di controllare determinate condizioni, come nell'esempio 7 la linea TEST controlla se sono stati mossi 9 byte. Se il registro X contiene il numero 9 vorrà dire che sono stati effettuati 9 movimenti.
Una istruzione di comparazione non cambia mai il contenuto di un registro o di una qualsiasi locazione di memoria. Perciò, il registro X non subisce nessuna variazione quando il 6502 esegue la linea TEST. Quello che potrebbe cambiare semmai sono i flag di stato del 6502.

I flag di stato

Oltre ai registri di uso comune (A, X e Y) il 6502 contiene uno speciale registro P detto anche *registro di stato*. I singoli bit del registro P vengono modificati ogni volta che il 6502 esegue determinate operazioni. Questi bit sono:

C bit 0: flag di Carry
Z bit 1: flag di Zero
I bit 2: flag di Interrupt
D bit 3: flag decimale

B	bit 4: flag di Break
	bit 5: non definito
V	bit 6: flag di Overflow
N	bit 7: flag negativo

In questo libro, non tratteremo l'uso di tutti i flag del registro P. In questo velocissimo corso sulla programmazione in Assembler e nel software presentato più avanti, i tre flag con cui avremo a che fare sono C, Z e N. Un'istruzione di comparazione (CMP, CPX, CPY) non influenza il valore contenuto nei registri A, X o Y, ma cambia lo stato dei flag C, Z e N. Per esempio, se un registro è comparato con un valore uguale, il flag Z viene settato (1), contrariamente viene azzerato (0). Se un'istruzione pone a 1 il settimo bit di un registro o di un indirizzo, anche il flag N viene settato, così pure se un'istruzione azzera il settimo bit di un registro o di un indirizzo, anche il flag N viene azzerato. Similmente, operazioni matematiche e logiche settano o azzerano il flag C che si comporta come il nono bit in tutte le operazioni logiche e aritmetiche. La Tabella 3.1 riassume gli effetti di un'istruzione di comparazione sul registro P.

Tabella 3.1 Effetti di un'istruzione di comparazione sui flag di stato. Notare che se si vuole testare lo stato del flag C dopo una comparazione è necessario settarlo (usando l'istruzione SEC) prima della comparazione. Quando si testa il flag N pensate agli input come valori di 8 bit con segno

Comparazione con valore *uguale*	C=1	N=0	Z=1
Comparazione con valore *maggiore*	C=0	N=0	Z=0
Comparazione con valore *minore*	C=1	N=0	Z=0

Salti condizionati

Testando lo stato dei flag possiamo far compiere al programma determinate azioni piuttosto che altre. Per esempio, le due istruzioni BEQ (*Branch EQual* - salta se uguale) e BNE (*Branch Not Equal* - salta se diverso) fanno in modo che il 6502 salti ad eseguire nuove istruzioni a seconda dello stato del flag Z. Una istruzione che permette al 6502 di saltare in base allo stato di un flag viene detta *istruzione di salto condizionato*. Altre istruzioni di salto condizionato dallo stato dei flag sono illustrate in Tabella 3.2.

Tabella 3.2 Istruzioni di salto condizionato

Flag	Istruzione	Descrizione	Opcode
C	BCC	salta se C=0	90
C	BCS	salta se C=1	B0
N	BPL	salta se risultato positivo	10
N	BMI	salta se risultato negativo	30
Z	BEQ	salta se uguale (Z=1)	F0
Z	BNE	salta se diverso (Z=0)	D0
V	BVC	salta se V=0	50
V	BVS	salta se V=1	70

La linea TEST dell'esempio 7 compara il registro X col valore 9; questo setta o azzera il flag Z. La linea SALTA controllando lo stato del flag Z ritorna alla linea LEGGE se il risultato della comparazione è diverso. Se Y fosse stato uguale a 9, il risultato della comparazione sarebbe stato uguale e il 6502 non sarebbe ritornato alla linea LEGGE.

Loop

L'esempio 7 viene tipicamente detto *loop*. È possibile fare in modo che il 6502 compia una certa operazione parecchie volte, inizializzando e incrementando un contatore e testandolo ogni volta per vedere se il lavoro è finito.

I loop sono molto potenti. Che cosa dovremmo aggiungere o cambiare nell'esempio 7 per fare sì che non 9, ma 90 byte vengano spostati da una locazione di memoria a un'altra? Non dovremo aggiungere niente fortunatamente, ma semplicemente cambiare l'operando della linea TEST. Invece di comparare il registro X con il valore 9, dovremo compararlo con 90, come si vede nell'esempio 8.

Esempio 8

Come muovere 90 byte consecutivi da una locazione ad un'altra

```
ORIGINE=$0200
DESTINAZIONE=$0300
```

```
INIZIO    LDX #0                  Inizializza il registro X
LEGGE     LDA ORIGINE,X           Legge l'X-esimo byte dal
                                  blocco di origine
SCRIVE    STA DESTINAZIONE,X      Scrive l'X-esimo byte nel
                                  blocco di destinazione
SOMMA     INX                     Incrementa il registro X di 1
TEST      CPX #90                 È il 90-esimo byte?
SALTA     BNE LEGGE               Se non è, legge il successivo
```

Usando i loop è possibile scrivere programmi non solo compatti, ma anche facilmente adattabili a particolari applicazioni. In ogni caso non sarebbe possibile usare i loop senza l'indirizzamento indicizzato e i salti. I loop possono essere anche insidiosi. Dov'è l'errore nel seguente loop?

Esempio 9

```
          ORIGINE=$0200
          DESTINAZIONE=$0300

INIZIO    LDX #0                  Inizializza il registro X
LEGGE     LDA ORIGINE,X           Legge l'X-esimo byte dal
                                  blocco di origine
SCRIVE    STA DESTINAZIONE,X      Scrive l'X-esimo byte nel
                                  blocco di destinazione
TEST      CPX #9                  È il nono byte?
SALTA     BNE LEGGE               Se non è, legge il successivo
```

Esaminate attentamente l'esempio 9. In che cosa differisce dall'esempio 7? Manca la linea SOMMA che incrementa il valore del registro X. Proviamo ad immaginare che cosa accade quando il 6502 esegue il programma dell'esempio 9. Nella prima linea il registro X è posto a 0; la seconda linea legge il byte dalla locazione \$0200+X (quindi \$0200) e la terza linea lo riscrive nella locazione \$0300+X (quindi \$0300). La quarta linea controlla il valore di X e torna alla linea LEGGE poiché X=0. Alla linea LEGGE il ciclo si ripeterà fino al TEST, ma, poiché in questo programma non c'è nulla che incrementa il valore di X, il TEST non verrà mai verificato e il loop sarà senza fine.
Diamo uno sguardo all'esempio 10. Ci sarà un loop? E se sì, sarà un loop finito o infinito? Perché sì o perché no?

Esempio 10

```
        ORIGINE=$0200
        DESTINAZIONE=$0300

INIZIO  LDX #0                 Inizializza il registro X
LEGGE   LDA ORIGINE,X          Legge l'X-esimo byte dal
                               blocco di origine
SCRIVE  STA DESTINAZIONE,X     Scrive l'X-esimo byte nel
                               blocco di destinazione
SOMMA   INX                    Incrementa il registro X di 1
TEST    CPX #9                 È il nono byte?
SALTA   BNE INIZIO             Se non è, legge il successivo
```

Indirizzamento relativo

Tutte le istruzioni di salto condizionato si avvalgono dell'indirizzamento relativo. Come l'indirizzamento in pagina zero (indicizzato e no) anche l'indirizzamento relativo richiede un solo byte nell'operando. L'operando specifica l'indirizzo relativo dell'opcode a cui il 6502 salterà se è verificata la condizione di test sul registro di stato. Una locazione relativa di 04 significa che l'opcode da eseguire si trova 4 byte più avanti del successivo opcode, naturalmente se la condizione è verificata, altrimenti il 6502 eseguirà l'opcode immediatamente successivo.
Poiché l'operando di un salto condizionato è lungo un solo byte, non sono ammessi salti in avanti maggiori di 127 byte e indietro maggiori di 128 byte. Si intende salto indietro quando il valore dell'operando è negativo, avanti quando è positivo. Un byte è negativo se ha il settimo bit settato mentre è positivo se il settimo bit è azzerato. Per questo motivo il valore 00 è considerato positivo. Esiste poi una istruzione chiamata JMP che permette di specificare un salto incondizionato a qualsiasi locazione di memoria; sarà perciò possibile indirizzare condizionatamente qualsiasi locazione di memoria, usando un salto condizionato a una locazione che contiene l'istruzione di salto incondizionato.

Salti incondizionati

Come in BASIC esiste l'istruzione GOTO che permette di saltare incondizionatamente a una linea di programma, anche in Assembler esiste l'istruzione JMP che permette di saltare senza condizioni a un indirizzo

specificato. È possibile far compiere al 6502 un loop infinito, semplicemente eseguendo un salto incondizionato alla locazione di inizio. Guardiamo l'esempio 11: fino a che una linea di programma all'interno del loop non causerà un salto a una locazione fuori dal loop, il 6502 continuerà ad eseguire le istruzioni in esso contenute.

Esempio 11

Loop infinito:

```
INIZIO  xxxxxxxxxx    alcune istruzioni
        xxxxxxxxxx        .
        xxxxxxxxxx        .
        JMP INIZIO
```

Indirizzamento indiretto

Un'istruzione JMP può essere usata sia con l'indirizzamento assoluto che con quello indiretto. L'indirizzamento assoluto è quello usato nell'esempio 11, dove l'operando rappresenta l'indirizzo di memoria a cui saltare. Ma nell'indirizzamento indiretto, che si indica contenendo l'operando tra parentesi, l'operando specifica l'indirizzo di un puntatore. Il 6502 salterà all'indirizzo specificato dal puntatore, non al puntatore stesso.
La linea JMP (puntatore) farà compiere al 6502 un salto alla locazione specificata dai byte contenuti nelle locazioni *puntatore* e *puntatore*+1. Perciò, se *puntatore*=$0600 e nelle locazioni $0600 e $0601 sono contenuti i valori $00 e $20, verrà eseguito un salto alla locazione $2000. (Ricordatevi che gli indirizzi sono memorizzati con prima il byte basso).

Funzionamento dei salti

Tutte le istruzioni di salto, relativo, assoluto o indiretto, influenzano il PC (*Program Counter* - contatore di programma). Prima che ogni istruzione di salto sia eseguita, il PC contiene l'indirizzo dell'opcode corrente. Una istruzione di salto modifica il valore contenuto nel PC in modo che nel ciclo macchina successivo, il 6502 non leggerà l'istruzione successiva, ma quella che si trova all'indirizzo specificato dall'istruzione di salto. Dopo un salto, l'elaborazione continua regolarmente dall'indirizzo raggiunto.

Rilocabilità

Generalmente è meglio implementare salti incondizionati come:

```
CLC
BCC    LOCA
```

piuttosto che:

```
JMP    LOCA
```

Il motivo è molto semplice. Il primo metodo, che fa uso dell'indirizzamento relativo piuttosto che assoluto, funzionerà sempre anche quando il programma sarà rilocato in altre aree di memoria. Scrivere i programmi in modo rilocabile porterà certamente ad un risparmio di tempo e di problemi nel momento in cui sarà necessario spostarli in differenti aree di memoria, pur mantenendo la loro efficienza.
Per rilocare in memoria il programma del secondo esempio è necessario cambiare l'operando, perché l'indirizzo assoluto LOCA sarà differente, mentre il programma del primo esempio non necessita di alcun cambiamento.

Subroutine

Probabilmente le due istruzioni più potenti disponibili in Assembler sono: JSR (*Jump to SubRoutine* - vai alla subroutine) e RTS (*ReTurn from Subroutine* - ritorna dalla subroutine). Queste istruzioni, che equivalgono in BASIC a GOSUB e RETURN, permettono di studiare dei blocchetti di programma, chiamati *subroutine*, usabili da diversi programmi.
Pensate a una subroutine come a un lavoro. Una volta insegnato al 6502 un lavoro, sarà inutile spiegarglielo una seconda volta. Prendiamo per esempio un programma in cui lo stesso lavoro deve essere eseguito parecchie volte. Dove il programma lo richiede, voi potete inserire le istruzioni per compiere il lavoro. D'altra parte, è possibile scrivere queste istruzioni, come subroutine, in una determinata area di memoria e chiamare questa subroutine dove richiesto dal programma principale. Una chiamata alla subroutine avrà così l'effetto di far eseguire al 6502 la subroutine e poi di ritornare al programma principale.
È necessaria una sola linea di programma per chiamare una subroutine. L'istruzione JSR SUB chiamerà la subroutine il cui indirizzo è SUB (come se fosse un salto incondizionato), il 6502 leggerà ed eseguirà le istru-

zioni contenute nelle locazioni successive alla SUB fino a che non incontrerà un RTS.
Quando viene letta l'istruzione RTS verrà attuato un salto di ritorno al programma chiamante e quindi all'istruzione immediatamente successiva alla JSR.
In pratica, quando una linea di programma chiama una subroutine, il 6502 ricorda dove si trovava prima del salto e quindi, quando viene incontrata l'istruzione RTS, il 6502 sa dove tornare proprio perché si ricorda l'indirizzo da dove è partito.
La Figura 3.1 illustra questo procedimento. Notate che la stessa subroutine può essere chiamata da punti differenti dello stesso programma e, immancabilmente, tornerà al punto di chiamata.
Le subroutine permettono di strutturare i programmi. Il vantaggio del software strutturato è quello di poter modificare parecchi programmi semplicemente cambiando una subroutine. Se, per esempio, tutti i programmi fanno uso della stessa subroutine di stampa, ogni volta che questa subroutine verrà migliorata, sarà migliore anche il comportamento di tutti i programmi che la usano. Inoltre è molto più facile modificare dei dati in un posto solo piuttosto che doverne modificare parecchi in posti differenti e normalmente non chiaramente documentati. Questo è il motivo per cui tutti i programmi presentati in questo libro fanno ampiamente uso di subroutine.

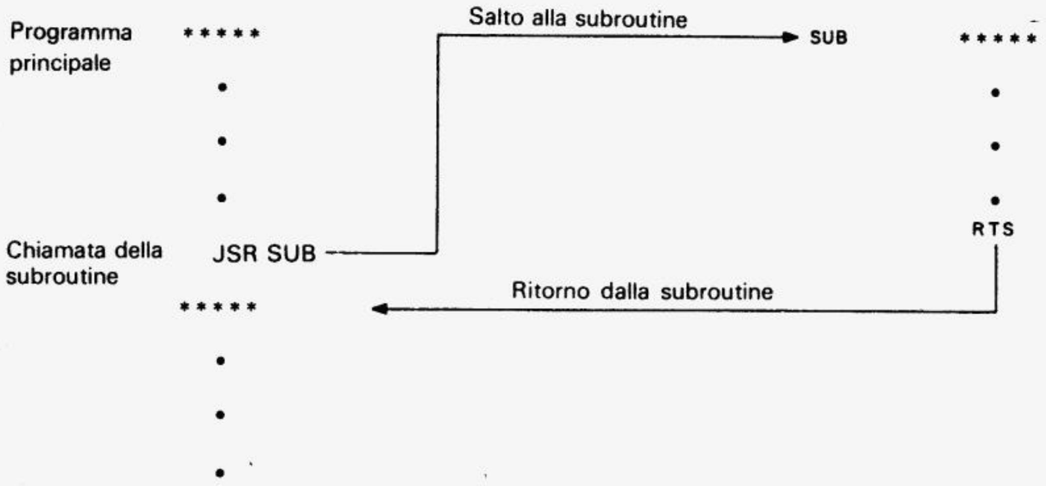


Figura 3.1 Salto e ritorno da una subroutine. Quando il microprocessore incontra una JSR, esegue la prima istruzione della subroutine. La subroutine SUB viene chiamata dal programma principale. L'ultima istruzione di una subroutine deve essere RTS che rende il controllo all'istruzione immediatamente seguente la chiamata. La subroutine SUB può essere chiamata da qualsiasi punto del programma dove è necessaria la particolare funzione che svolge

Subroutine fittizie

Una subroutine fittizia non contiene altro che un'istruzione RTS. Una chiamata a una subroutine fittizia non modificherà nulla perché il 6502 ritornerà immediatamente al programma principale con i registri intatti. La ragione di una chiamata a una subroutine fittizia è quella di fornire un aggancio usabile in un secondo tempo per sostituire alla subroutine fittizia quella effettiva. Nello sviluppare un programma sarà possibile inserire delle chiamate alla subroutine fittizia e in seguito, dopo avere generato le subroutine necessarie, sostituire l'indirizzo della subroutine fittizia con quello della subroutine effettiva. Cercare di inserire una chiamata in un programma che non prevede questi agganci vi potrebbe far sognare un "calzascarpe per memoria" che permettesse di inserire 3 byte extra di programma nella stessa area di memoria.

Lo stack

Oltre ai modi di indirizzamento che permettono al 6502 di accedere alla memoria, esiste un modo che permette al 6502 l'accesso a una particolare area di memoria, lunga 256 byte, chiamata *stack* (catasta).
Possiamo immaginare lo stack come una pila di mattoni. L'unico modo possibile di aggiungere un mattone alla pila è quello di posarlo sopra tutti gli altri come pure per prenderne uno dovremo partire dal più alto. Questo metodo viene comunemente detto LIFO (*Last In First Out* - l'ultimo che entra è il primo a uscire). L'ultimo mattone posto sulla pila sarà il primo ad essere rimosso.
Nel nostro caso chiameremo *push* l'operazione corrispondente a porre un mattone sulla pila e *pop* l'operazione corrispondente a toglierne uno. L'ultimo elemento della pila è il *top* (cima) dello stack.
Pensiamo, per esempio, di porre due byte sullo stack (ogni byte ha un valore di 8 bit, forse un numero o un carattere ASCII; vedi Figura 3.2a).
Per prima cosa eseguiremo un push del primo byte, come si vede in Figura 3.2b. Tutte le locazioni sopra il primo byte vengono dette "vuote" e il primo byte è la cima dello stack.
Ora facciamo un push nello stack con il secondo byte (Figura 3.2c). Cosa è successo? Il secondo byte adesso è la cima dello stack mentre il primo byte è ugualmente contenuto nello stack. Se togliamo il secondo byte dallo stack, eseguiamo un pop (Figura 3.2d). Il primo byte adesso è di nuovo la cima dello stack. Un altro pop toglierà anche il primo byte lasciando lo stack completamente vuoto. Notate che è stato necessario levare il secondo byte prima di poter togliere anche il primo. Questo è il procedimento chiamato LIFO.

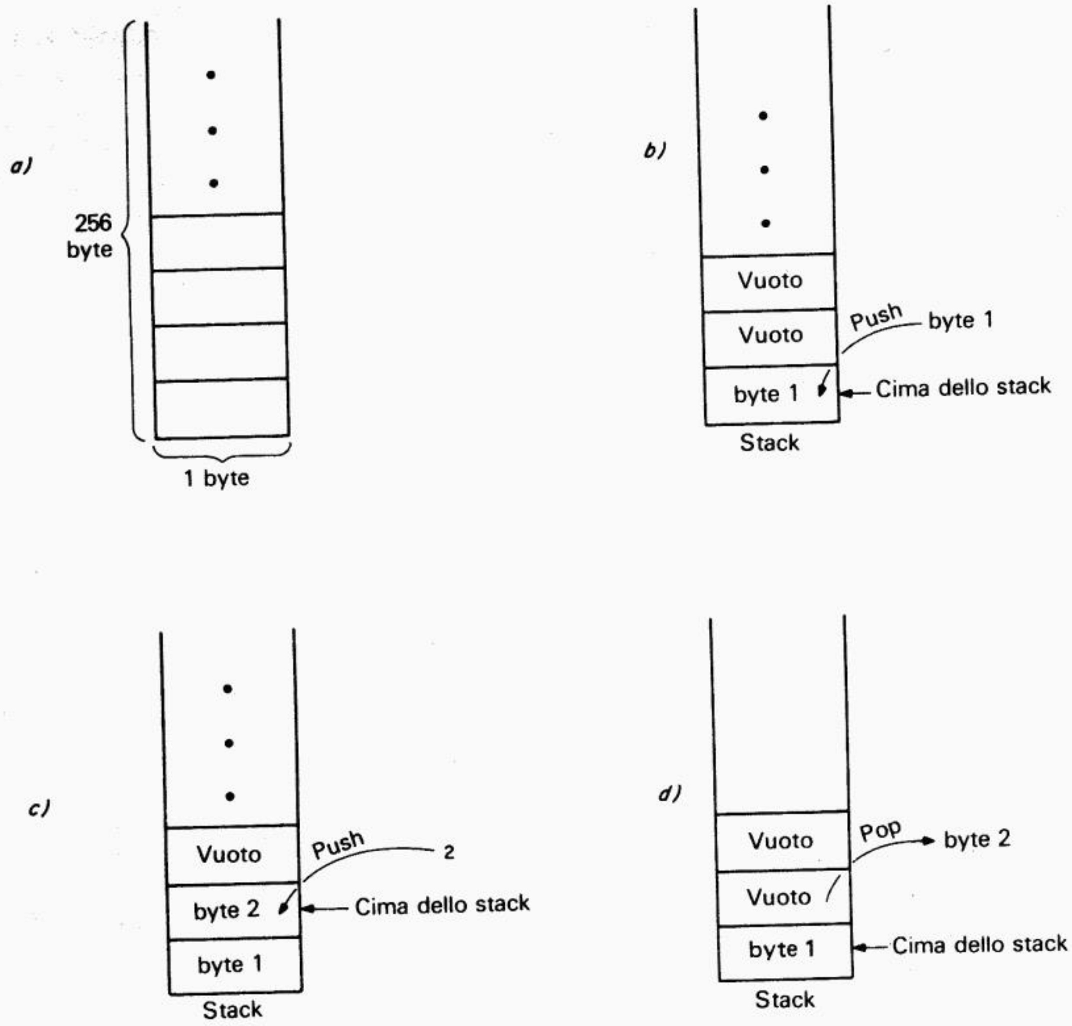


Figura 3.2 L'uso di push e pop

L'istruzione PHA (*PusH A*) permette di inserire il valore dell'accumulatore nello stack, mentre l'istruzione PLA (*Pop and Load Accumulator*) permette di caricare l'accumulatore con il valore contenuto nella cima dello stack. L'istruzione PHP (*PusH P*) permette di trasferire il registro P nello stack, mentre la funzione contraria viene svolta dall'istruzione PLP (*Pop Load P*).
Lo stack è un'area di memoria molto pratica quando si presenta la necessità di memorizzare temporaneamente alcuni byte, senza dover ricorrere all'uso di locazioni assolute. Le subroutine possono trasferire i dati ai programmi principali per mezzo dello stack ma, se una subroutine scrive dei dati nello stack e poi fallisce nella lettura di quei dati prima dell'RTS, non avverrà il ritorno al programma chiamante.

Questo avviene perché il 6502, quando incontra una JSR, memorizza l'indirizzo a cui tornare proprio nello stack. Una subroutine può tornare al programma principale solo perché l'indirizzo di ritorno è memorizzato nello stack e, se questo indirizzo non si trova in cima allo stack quando viene eseguita l'istruzione RTS, la subroutine non tornerà al programma chiamante. Così nelle subroutine sarà sempre necessario resettare lo stack prima di incontrare l'RTS.

Capitolo

Aritmetica e logica 4

Transcodifica dei caratteri

Come dimostrato nell'esempio 7, l'indirizzamento indicizzato è utile quando si deve compiere una determinata operazione, come ad esempio un trasferimento su una serie contigua di byte. Esiste però un'altra importante operazione che è possibile compiere e riguarda le tabelle di confronto. Diciamo, per esempio, che voi e un vostro amico avete deciso di inviarvi dei messaggi usando un codice di sostituzione. Ad ogni numero, lettera o segno di interpunzione, vi siete accordati di sostituire un carattere differente; una "A" sarà sostituita con una "R", un ";" con un "9", ecc. Sia voi che il vostro amico avete una tabella che riporta per ogni carattere il corrispondente da sostituire. Quando scrivete un messaggio, quindi, lo transcodificate usando la tabella e lo inviate al vostro amico. Chiunque verrà in possesso del messaggio difficilmente sarà in grado di comprenderne il contenuto, ma il vostro amico, facendo uso della tabella, riuscirà a decodificarlo.
Molto spesso i programmi richiedono una decodifica per i caratteri ed è possibile compiere questo lavoro sfruttando le tabelle di codifica/decodifica.

Esempio 12

Subroutine di transcodifica

```
TRANS   TAX             Usa il carattere da transcodificare come indice della tabella
        LDA TABELLA,X   Legge nella tabella il valore corrispondente
        RTS             Ritorna al programma principale
                        A contiene il carattere transcodificato
                        X contiene il carattere originale
```

Trasferimento dei registri

La subroutine TRANS, nell'esempio 12, assume che il byte da transcodificare sia contenuto nell'accumulatore. Questo byte può essere una lettera, un numero, un segno d'interpunzione, un codice di controllo, un carattere grafico, ma in ogni caso sarà un valore di otto bit. La prima linea di TRANS sposta il valore dell'accumulatore nel registro X per mezzo dell'istruzione TAX.
Le istruzioni di trasferimento interagiscono solo con i registri e in nessun modo possono modificare la memoria indirizzabile. Queste istruzioni permettono di trasferire un valore da un registro ad un altro. Dopo avere effettuato una operazione di trasferimento, il registro di partenza resterà inalterato, mentre quello di arrivo conterrà lo stesso valore del registro di partenza. Le istruzioni di trasferimento sono:

```
TAX   trasferisce A in X
TAY   trasferisce A in Y
TXA   trasferisce X in A
TYA   trasferisce Y in A
```

Il trasferimento dei registri modifica i flag N e Z. Queste istruzioni permettono di trasferire valori dall'accumulatore ai registri X o Y oppure di trasferire il valore contenuto nei registri X o Y all'accumulatore. Come faremo a trasferire un valore da X a Y o viceversa? (Suggerimento: occorrono due linee di programma e ogni linea contiene un'istruzione elencata qui sopra).

Tabelle di transcodifica

La seconda linea dell'esempio 12 è quella che si occupa della transcodifica cercando il dato richiesto nella tabella. La TABELLA contiene l'indirizzo di base di una tabella che abbiamo preventivamente introdotto in memoria. L'indirizzamento indicizzato permette di leggere l'X-esimo byte a partire dall'indirizzo di base (cioè l'X-esimo byte della tabella). Una volta eseguita questa operazione, la ricerca è finita; il 6502 contiene nell'accumulatore l'X-esimo byte della tabella e deve solo ritornare al programma principale. Questo si ottiene con la terza linea della subroutine che non può essere altro che un RTS.
A questo punto è possibile ottenere una transcodifica all'interno del programma principale semplicemente eseguendo:

```
JSR TRANS
```

L'uso delle tabelle di transcodifica permette di avere programmi molto flessibili. Se un programma fa uso di tabelle e si vogliono ottenere funzioni differenti, sarà probabilmente necessario cambiare solo i valori delle tabelle. Le tabelle sono veloci e flessibili, ma presentano uno svantaggio. Infatti, possiamo avere tutte le tabelle necessarie per permettere alla macchina di cercare la risposta voluta direttamente in memoria, ma se la memoria non è sufficiente, il computer sarà costretto a calcolarsi le risposte.

Operazioni aritmetiche

Il 6502 può eseguire le seguenti operazioni aritmetiche a 8 bit:

traslazione (*shift*)
rotazione (*rotate*)
incremento
decremento
somma
sottrazione

Per comprendere più facilmente le operazioni che il 6502 compie su di un byte è necessario pensare allo stato dei singoli bit di quel byte.
Anche se il byte rappresenta un numero o una lettera, non pensate a quello che potreste fare con quel numero o lettera, ma pensate alla configurazione dei bit. Cosa possiamo combinare con i bit?

TRASLAZIONE

È possibile traslare i bit che compongono un byte di una posizione a sinistra o a destra. L'istruzione ASL (*Arithmetic Shift Left* - traslazione aritmetica a sinistra) manipola un byte muovendo ogni bit a sinistra di una posizione. Il bit più a sinistra (bit 7) viene trasferito nel Carry (flag C) e il bit più a destra viene azzerato come si vede in Figura 4.1.

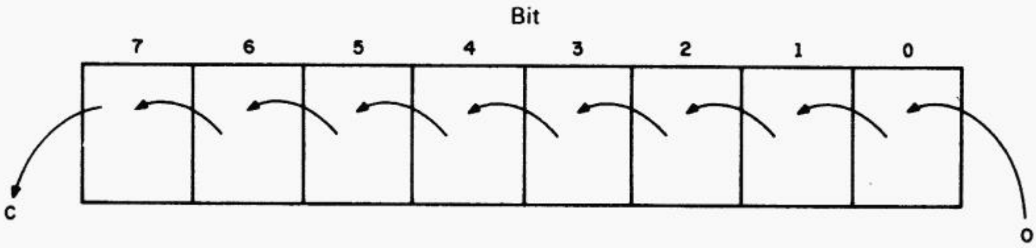


Figura 4.1 Effetto dell'istruzione ASL

Prendiamo, per esempio, un byte alla locazione TMP con la seguente configurazione:

TMP   0 1 0 1 0 1 1 0

Dopo un'istruzione ASL TMP il byte avrà questa configurazione:

TMP   1 0 1 0 1 1 0 0

con il flag C (Carry) contenente lo stato del bit che si trovava più a sinistra (nel nostro caso). Se eseguiamo ancora la stessa istruzione il byte diventerà:

TMP   0 1 0 1 1 0 0 0

e il flag C sarà settato a 1.

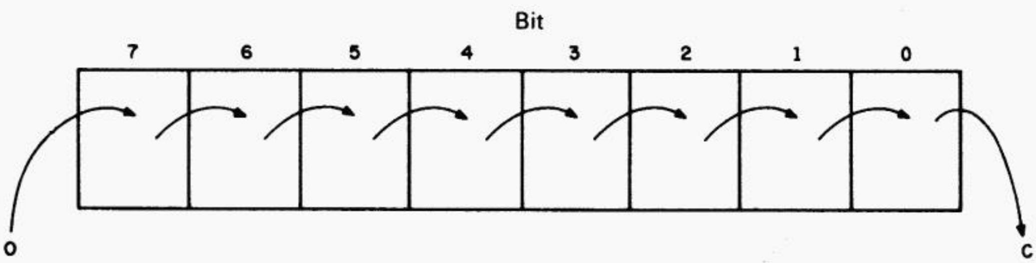


Figura 4.2 Effetto dell'istruzione LSR

L'istruzione LSR (*Logical Shift Right* - traslazione logica a destra) ha esattamente l'effetto contrario. Tutti i bit vengono traslati a destra verso il flag C e il bit a sinistra viene settato a 0. Questo processo è illustrato in Figura 4.2.
Se teniamo come esempio il byte alla locazione TMP con la configurazione data in origine, dopo un'istruzione LSR TMP il dato si presenterà così:

TMP 0 0 1 0 1 0 1 1

con il flag C che rispecchia lo stato del bit che stava più a destra, nel nostro caso 0. Se la stessa istruzione viene eseguita di nuovo, il dato diventerà:

TMP 0 0 0 1 0 1 0 1

e il flag C sarà posto a 1.
Poiché in binario abbiamo la rappresentazione di un numero (ogni bit rappresenta una potenza successiva del 2), alcune operazioni aritmetiche diventano molto semplici. Per dividere un byte per due sarà sufficiente traslare a destra, mentre per moltiplicare il valore in un byte per due sarà necessario operare una traslazione a sinistra.

ROTAZIONE

È possibile anche ruotare i bit di un byte a sinistra o a destra attraverso il flag C. Diversamente dalla traslazione, la rotazione conserva tutte le informazioni originalmente contenute nel byte.
La Figura 4.3 mostra il funzionamento dell'istruzione ROL (*ROtate Left*-rotazione a sinistra). Per esempio, consideriamo il byte della locazione TMP come negli esempi precedenti e diciamo che il flag C è settato (C=1):

TMP 0 1 0 1 0 1 1 0

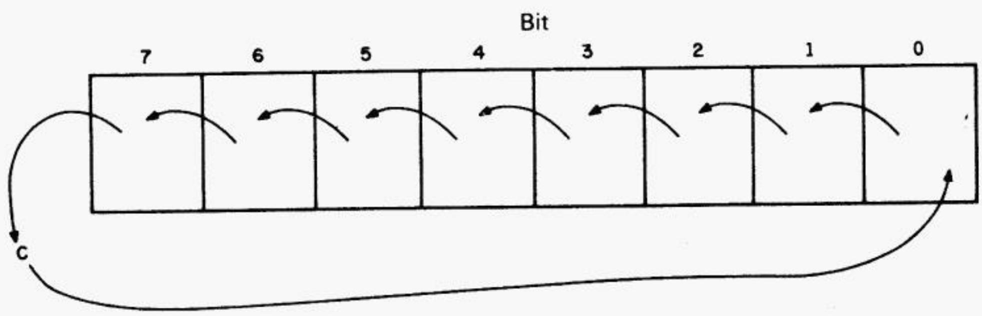


Figura 4.3 Effetto dell'istruzione ROL

Dopo l'istruzione ROL, TMP il byte diventa:

TMP 1 0 1 0 1 1 0 1

e il flag C verrà posto allo stato del bit più a sinistra. Nel nostro caso, C sarà a 0 e il bit sarà a 1. A parte questo comportamento di C e del bit 0 tutto il resto è molto simile all'istruzione ASL. Dopo aver eseguito una seconda volta un'istruzione ROL TMP il byte si presenterà così:

TMP 0 1 0 1 1 0 1 1

e C sarà a 1.
In un'istruzione ROL il bit 0 assume sempre lo stesso valore di C (mentre con ASL il bit 0 viene sempre azzerato). Se fosse stata eseguita una istruzione ASL quale sarebbe stata la configurazione del bit di TMP?
La Figura 4.4 mostra il funzionamento dell'istruzione ROR (*ROtate Right*-rotazione a destra). Questa istruzione è simile alla ROL; la differenza sta nel fatto che il bit 7 viene settato da C e C viene settato dal bit 0.

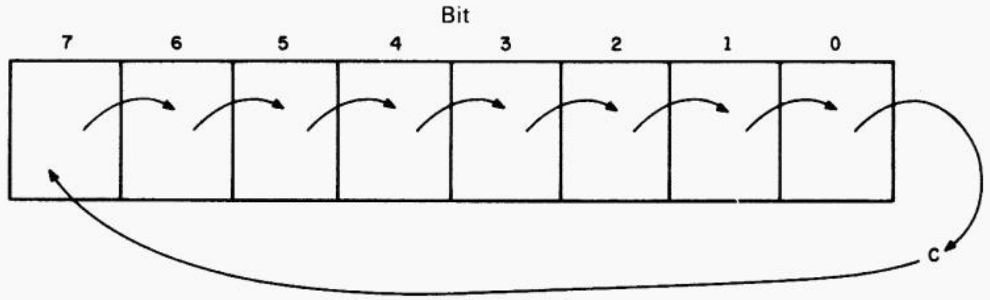


Figura 4.4 Effetto dell'istruzione ROR

Ruotando un byte a destra o a sinistra nove volte consecutive otterremo come risultato lo stesso byte di partenza. Traslando invece un byte nove volte di seguito ci ritroveremo con un byte completamente azzerato, cioè con niente.

INCREMENTO E DECREMENTO

È possibile incrementare o decrementare un byte in tre modi differenti: usando le istruzioni INC e DEC quando si opera su una locazione di memoria, usando INX e DEX quando si opera sul registro X e con INY e DEY quando si opera sul registro Y. Nessuna di queste istruzioni modifica il flag C, mentre viene ad essere influenzato il flag Z. Z viene settato

se il risultato dell'operazione di incremento o decremento è zero, mentre viene azzerato se l'operazione dà come risultato un valore diverso da zero. Il flag N viene settato se il byte risultante dopo un incremento o decremento ha il bit 7 a 1; altrimenti N viene azzerato.
È da notare che se si incontra un registro o un byte il cui valore è $FF, il risultato sarà $00. Similmente se si decrementa un registro o un byte di valore $00 il risultato sarà $FF.
Non è possibile incrementare o decrementare l'accumulatore, ma è possibile sommargli o sottrargli un byte.

SOMMA

L'esempio 13 mostra come sommare all'accumulatore il byte contenuto nella locazione NUMERO:

Esempio 13

CLC	Azzera il flag C
ADC NUMERO	Somma all'accumulatore il byte contenuto nella locazione NUMERO

Dopo l'esecuzione di queste istruzioni, l'accumulatore conterrà gli otto bit bassi del risultato della somma. Se, a seguito della somma, il flag C viene settato, significa che il risultato è un valore maggiore di 255; se il flag C resta a zero, il risultato della somma è un valore minore di 256, e perciò l'accumulatore contiene il vero risultato dell'operazione. È sempre necessario azzerare il flag C prima di eseguire l'istruzione ADC (*ADd with Carry* - somma con il riporto).

SOTTRAZIONE

La sottrazione è un'operazione facile come la somma. Per sottrarre un byte dall'accumulatore, è prima di tutto necessario settare il flag C con l'istruzione SEC (*SEt Carry* - setta C) e poi sottrarre dall'accumulatore una costante o il valore contenuto in una locazione di memoria, per mezzo dell'istruzione SBC (*SuBtract with Carry* - sottrae con il riporto):

SEC	Setta C
SBC OPER	Sottrae dall'accumulatore il valore contenuto nella locazione OPER

Se il valore dell'operando è maggiore del valore contenuto nell'accumulatore, il flag C viene azzerato altrimenti resta settato. In ogni caso l'accumulatore conterrà il risultato a 8 bit.
Ricapitolando, dobbiamo settare C prima di sottrarre e azzerarlo prima di sommare. Se il flag C non cambia di stato, l'accumulatore contiene il risultato completo dell'operazione; ma se avviene un cambiamento di stato allora significa che il risultato è maggiore di 255 (per l'addizione) e minore di 0 (per la sottrazione).

Modo decimale

Il registro di stato, P, contiene un bit chiamato *flag decimale*. Se questo flag è settato, il 6502 esegue le addizioni e le sottrazioni in modo decimale, altrimenti in modo binario. Operare in modo decimale significa che i byte sono trattati in BCD (*Binary Coded Decimal* - decimale codificato binario) e che i 4 bit bassi possono rappresentare un valore compreso tra 0 e 9 come pure i 4 bit alti. Nessun nybble (4 bit) può contenere valori compresi tra A ed F; quindi ogni nybble rappresenta un numero decimale.
Le istruzioni SED (*SEt Decimal* - setta D) e CLD (*CLear Decimal* - azzera D) sono usate rispettivamente per settare e azzerare il flag decimale. Noterete che tutto il software presentato in questo libro non fa uso del modo decimale.
Il valore decimale 255 è il numero più grande che siamo in grado di rappresentare in BCD, ma il numero 99 è il valore più grande rappresentabile da un byte in BCD.

Operazioni logiche

Come si può cambiare lo stato di un singolo bit in un byte senza alterare lo stato degli altri? Situazioni di questo genere sono usuali durante le operazioni di I/O, e possono essere svolte per mezzo delle istruzioni ORA, AND e XOR.

SETTARE I BIT

L'istruzione ORA ci permette di settare uno o più bit dell'accumulatore senza influenzare lo stato degli altri. ORA esegue l'operazione logica OR tra l'accumulatore e un byte specificato detto anche *maschera*. Il bit n dell'accumulatore sarà settato se è inizialmente settato oppure se lo è il bit n della maschera oppure ancora se entrambi i bit n sono settati. Men-

tre il bit n sarà azzerato solo se entrambi i bit n dell'accumulatore e della maschera saranno azzerati. La Tabella 4.1 rappresenta la tabella della verità dell'operazione logica OR.
Una tabella della verità ci dà tutte le possibili combinazioni di due bit su cui si può operare e il risultato dell'operazione che si compie (in questo caso OR).

Tabella 4.1 Tabella della verità dell'operando OR

Bit 1		Bit 2	Risultato
0	OR	0	0
0	OR	1	1
1	OR	0	1
1	OR	1	1

Supponiamo per esempio di eseguire l'istruzione ORA #$80. In questo caso la maschera è $80, rappresentabile anche come 10000000. Questa istruzione setterà il bit 7 dell'accumulatore lasciando inalterato lo stato degli altri bit. Quindi, se l'accumulatore prima dell'operazione conteneva il valore 00010010, dopo l'operazione conterrà 10010010.
Un altro esempio potrebbe essere ORA #3. Poiché il valore decimale 3 viene convertito nella maschera binaria 00000011 ne deduciamo che l'operazione setterà i due bit bassi dell'accumulatore lasciando inalterati i bit da 2 a 7.
Che valore dovremo scegliere per settare il nybble alto dell'accumulatore? E per quello basso?

AZZERARE I BIT

Usando l'istruzione AND è possibile azzerare uno o più bit dell'accumulatore senza alterare lo stato degli altri. AND compie l'operazione logica AND tra l'accumulatore e un byte di maschera specificato nell'operando. L'istruzione AND setterà il bit n dell'accumulatore solo se il bit n dell'accumulatore è settato e se lo è pure il bit n della maschera. Se il bit n dell'accumulatore è azzerato o se lo è il bit n della maschera, allora, dopo l'operazione, il bit n dell'accumulatore sarà azzerato.
La Tabella 4.2 è la tabella della verità dell'operazione logica AND.
Per esempio, l'istruzione AND #1 azzererà tutti i bit dell'accumulatore eccetto il bit 1 che resterà invariato. AND #$F0 azzererà il nybble basso dell'accumulatore lasciando inalterato quello alto. Scegliendo la masche-

Tabella 4.2 Tabella della verità dell'operando AND

Bit 1		Bit 2	Risultato
0	AND	0	0
0	AND	1	0
1	AND	0	0
1	AND	1	1

ra giusta, sarà quindi possibile azzerare qualsiasi bit o gruppo di bit dell'accumulatore senza influenzare gli altri.

COMMUTARE I BIT

L'istruzione XOR, OR esclusivo, permette di commutare uno o più bit dell'accumulatore lasciando inalterati gli altri. Compiendo uno XOR setteremo il bit n dell'accumulatore se il bit n dell'accumulatore è settato e se il bit n della maschera è azzerato o viceversa. Il bit n verrà azzerato se ha lo stesso stato sia nell'accumulatore che nella maschera. La Tabella 4.3 è la tabella della verità dell'operazione XOR.

Tabella 4.3 Tabella della verità dell'operando XOR (OR esclusivo)

Bit 1		Bit 2	Risultato
0	XOR	0	0
0	XOR	1	1
1	XOR	0	1
1	XOR	1	0

Per commutare il bit n dell'accumulatore è sufficiente compiere uno XOR fra l'accumulatore e una maschera che ha solo il bit n settato. Il bit n dell'accumulatore cambierà stato.
Le operazioni logiche, in congiunzione con le istruzioni di salto relativo, rendono possibile la decisione delle operazioni da eseguire testando lo stato di un singolo bit in memoria. Supponiamo per esempio, che sia necessario che il programma compia determinate operazioni (operazione A) se il bit 6 di un byte alla locazione FLAG è settato e compia altre operazioni (operazione B) se il bit è azzerato. L'esempio 14 mostra il modo di controllare lo stato del bit 6 di FLAG pur mantenendo il byte inalterato.

Esempio 14

```
         LDA FLAG        Legge il byte FLAG
         AND #$40        Azzera tutti i bit tranne il 6
         BEQ AZIONB
AZIONA   xxxxxxxxx
         xxxxxxxxx       Esegue l'azione A perché il bit 6
                         è settato
           :
AZIONB   yyyyyyyyy
         yyyyyyyyy       Esegue l'azione B perché il bit 6
                         è azzerato
```

Facciamo ora un esempio sull'utilità dei flag. In America sulle cassette postali situate davanti a ogni villetta c'è una bandierina di lamiera che può essere sollevata se c'è posta da prelevare o abbassata se non c'è posta. Va spiegato che nell'immensa provincia americana dove le abitazioni sono molto disperse sul territorio, il servizio postale prevede anche il ritiro a domicilio della posta in partenza. L'azione di alzare o abbassare questa bandierina non richiede molta fatica, e permette al postino di compiere il suo giro più velocemente, visto che ad ogni cassetta non dovrà scendere dal furgone per controllare la presenza o meno della posta. L'uso di questo semplice indicatore, che funziona esattamente come i flag nei linguaggi di programmazione, ottimizza immensamente il servizio. Il tran-tran del postino è, a tutti gli effetti, un esempio di programmazione molto sofisticata. Proviamo a pensare al postino come ad una persona che segue scrupolosamente un programma e analizziamo tutti i vantaggi offerti dall'uso dei flag.

Il "programma" postino si divide in due parti distinte: il lavoro che deve essere fatto in ufficio e il lavoro che deve essere fatto per strada. In ufficio il postino mette in ordine le varie lettere e le divide in sacchi destinati alle varie zone della città. L'ordinamento che avviene in ufficio permette al postino di compiere il suo giro più velocemente, senza dover ulteriormente, una volta sulla strada, fare ricerche o selezioni. Nella seconda parte del "programma" del postino abbiamo tutte le operazioni che devono essere compiute sulla strada.

Il postino carica la posta da distribuire sul furgone e comincia il suo giro. Guidando per la strada vede una cassetta delle lettere e pensa: c'è posta per le persone che abitano qui? Se la risposta è affermativa, il postino si ferma e inserisce la posta nella cassetta, ma cosa accade se non c'è posta per quell'indirizzo? È necessario fermarsi o proseguire per il prossimo indirizzo?

Il postino guarda meglio la cassetta. Come è la bandierina? Se è giù allo-

ra non c'è posta da ritirare e quindi può proseguire, altrimenti è costretto a fermarsi per ritirare la posta in partenza.
Un flag non è altro che un'informazione fornita da un singolo bit ma, interpretando lo stato di numerosi flag, perfino un semplice programma è in grado di fronteggiare situazioni assai complesse.
Se il vostro computer dispone di 8000 byte di memoria significa che ci sono 64000 bit di memoria. Sarebbe quindi possibile usare gran parte di questi bit come flag, per esempio simulando la presenza della posta in partenza in una comunità di oltre 50000 individui.
Comunque, non avete comperato un computer per giocare al postino e a questo punto avete abbastanza conoscenze per poter seguire e capire i programmi che saranno presentati nei capitoli successivi.
Questi programmi includono vari esempi per tutte le istruzioni e le tecniche di programmazione presentate in questo rapidissimo corso di linguaggio Assembler.
Il software contenuto nei prossimi capitoli vi fornirà inoltre alcuni strumenti utili allo sviluppo dei vostri programmi.
Un'ultima informazione prima di passare alle applicazioni pratiche: esiste un'istruzione che non provoca nulla; l'istruzione NOP (*NO oPeration* -nessuna operazione). Quale è il motivo di un'istruzione del genere? Occasionalmente può capitare di dover sostituire un'istruzione non voluta con una fittizia. Quando volete annullare parti di un programma senza doverlo riscrivere completamente potete cambiare le istruzioni non volute con dei NOP che vengono rappresentati in memoria come $EA.

Capitolo

Utility per la gestione dello schermo

5

Pensiamo ora a come visualizzare qualcosa sullo schermo. Nel C-64 e nel VIC-20 il circuito video scandisce una particolare area di memoria chiamata memoria video. Ogni indirizzo della memoria video rappresenta una posizione sullo schermo (da qui la definizione *video mappato in memoria*). Per ogni carattere contenuto nella memoria video il circuito di gestione visualizza il corrispondente carattere in una determinata posizione sullo schermo. Per visualizzare un carattere sullo schermo, sarà quindi necessario memorizzare il carattere desiderato nella memoria video e nella particolare locazione che corrisponde alla posizione di schermo. Per conoscere l'indirizzo corrispondente a una determinata posizione sullo schermo bisogna consultare la mappa della memoria video. Le Appendici B1 e B2 contengono le mappe della memoria video del C-64 e del VIC-20. È da notare che sistemi differenti avranno differenti locazioni per la stessa posizione fisica sullo schermo.
Piuttosto che indirizzare la memoria video in maniera assoluta, sarebbe più pratico indirizzarla in maniera indiretta, come se avessimo una freccia controllata via software in grado di spostarsi e indicare tutte le posizioni del video. Potremmo così individuare un carattere con la freccia, muoverlo e posizionarlo dove vogliamo, senza essere legati alla sua posizione assoluta.
Una cosa del genere è facilmente implementabile usando un puntatore in pagina zero.

Puntatori

Un puntatore non è altro che una coppia di byte consecutivi in memoria. Poiché un byte è formato da 8 bit, un puntatore è formato da 16 bit, il che significa che un contatore può specificare una delle 65536 ($=2^{16}$) differenti locazioni di memoria.
Un puntatore può specificare, o meglio puntare, solo un indirizzo alla volta. Il byte basso di un puntatore contiene gli 8 bit meno significativi (LSB) dell'indirizzo, mentre il byte alto contiene gli 8 bit più significativi (MSB) dell'indirizzo che specifica.
Ad esempio, se vogliamo un puntatore alla locazione $1000, dobbiamo riservare 2 byte per il puntatore, il che significa occupare i byte alle locazioni $1000 e $1001. $1000 conterrà il byte basso e $1001 quello alto. Se, per esempio, vogliamo che questo puntatore specifichi l'indirizzo $ABCD, dobbiamo fare come segue:

```
PUNTA=$1000                     Questa istruzione dell'Assembler as-
                                segna alla label PUNTA il valore
                                $1000
LDA #$CD        A9  CD          Setta il
STA PUNTA       8D  00  10      byte basso
LDA #$AB        A9  AB          Setta il
STA PUNTA+1     8D  01  10      byte alto
```

Adesso PUNTA specifica l'indirizzo $ABCD.
Anche se un puntatore può essere dovunque in memoria, è particolarmente potente quando si trova in pagina zero (gli indirizzi compresi da $0000 a $00FF). Il modo di indirizzamento indiretto permette a un puntatore in pagina zero di specificare l'indirizzo in cui certe operazioni devono essere svolte. Un puntatore in pagina zero deve essere situato ovviamente in pagina zero, ma può specificare qualsiasi locazione di memoria. Per esempio, un puntatore in pagina zero può essere usato per specificare gli indirizzi in cui i dati saranno memorizzati. Poiché la memoria video è come qualsiasi altra memoria ad accesso casuale, possiamo implementare la nostra freccia come un puntatore in pagina zero.

TV.PTR

Quello che ci occorre è un puntatore in pagina zero che punti ad una particolare locazione dello schermo. Quando vorremo esaminare o modificare il contenuto del video lo faremo attraverso questo puntatore che chiameremo TV.PTR.

Siccome sia il C-64 che il VIC-20 non fanno uso delle locazioni comprese tra $00FB e $00FE, faremo uso, per il nostro TV.PTR, delle locazioni $00FB e $00FC. Questo sarà possibile con la seguente istruzione Assembler:

```
TV.PTR=$FB
```

TV.SCR

Il puntatore TV.PTR determina la locazione corrente sullo schermo. Per visualizzare qualcosa nell'attuale posizione occorrerà semplicemente caricare l'accumulatore con un byte ed eseguire le seguenti linee di programma:

```
LDY #0          A0  00
STA (TV.PTR),Y  91  FB
```

Questa procedura ci permette di visualizzare qualcosa sullo schermo, ma cosa dovremo fare per visualizzare un dato carattere? Il codice ASCII di un carattere non è necessariamente uguale al codice video usato per rappresentarlo sullo schermo. Per visualizzare una "A" non possiamo semplicemente caricare l'accumulatore con il codice ASCII della A (che è $41) ed eseguire le linee riportate più sopra, perché il simbolo "A" ha un differente codice video sul C-64.
Come si può, allora, visualizzare un certo carattere ASCII? Ciò è possibile assumendo che esista una subroutine, chiamata FIXCAR, che trasformi ogni carattere del codice ASCII nel corrispondente codice video. FIXCAR sarà naturalmente differente da computer a computer e non è questo il luogo per entrare in ulteriori dettagli (vedere le appendici relative al vostro computer per il listato e la spiegazione di FIXCAR).
Diamo per scontato che esista la subroutine FIXCAR e che se viene chiamata dal programma principale con il carattere ASCII contenuto nell'accumulatore, ci restituisca il corrispondente codice video sempre nell'accumulatore.
Prima potevamo visualizzare un'immagine sullo schermo; con FIXCAR adesso siamo in grado di visualizzare il carattere ASCII che ci interessa. Siccome questa necessità si presenterà molto spesso, svilupperemo qui di seguito, una subroutine, TV.SCR, che ci permetterà di scrivere i caratteri sullo schermo:

```
TV.SCR   JSR FIXCAR        Converte il carattere ASCII nel co-
                           dice usato dal sistema
         LDY #0            Visualizza il carattere nella
         STA (TV.PTR),Y    posizione corrente
         RTS
```

La posizione sullo schermo

Questi esempi che ci permettono di modificare ed esaminare lo schermo funzionano solo se il puntatore TV.PTR punta realmente a queste posizioni. Perciò, prima di far funzionare i programmi degli esempi, dovremo essere certi che TV.PTR indirizzi alla locazione desiderata della memoria video.

Ci sono parecchi sistemi per ottenere questo risultato. Se vi interessa scrivere un programma specifico per un solo tipo di computer, o su più modelli che hanno la memoria video mappata alla stessa maniera, sarà possibile usare l'indirizzamento immediato per far sì che TV.PTR punti un determinato indirizzo della memoria video. Diciamo, per esempio, di voler settare TV.PTR in maniera che indirizzi la terza colonna della quarta riga (contando a destra e verso il basso da un'origine situata nell'angolo superiore sinistro dello schermo). Se siete in possesso di un VIC-20 provvisto dell'espansione ad 8K, consultando la documentazione troverete che l'indirizzo $1044 della memoria video corrisponde alla posizione che ci interessa. $10 è il byte alto e $44 quello basso; sarà quindi possibile predisporre TV.PTR con le seguenti linee di codice:

```
LDA #$44       A9  44   Setta il
STA TV.PTR     85  FB   byte basso
LDA #$10       A9  10   Setta il
STA TV.PTR+1   85  FC   byte alto
```

Queste istruzioni vengono eseguite velocemente ed hanno l'indubbio vantaggio della rilocabilità; ma non è molto conveniente dover accedere ad un indirizzo della memoria video ogni volta che si presenta la necessità di visualizzare qualcosa. Sarebbe molto più pratico poter indirizzare lo schermo come una coppia di coordinate X e Y. Perché non creare una subroutine che, prendendo le coordinate X e Y, inizializzi per noi TV.PTR?

TVTOXY

TVTOXY è una subroutine che inizializza il puntatore TV.PTR in modo che punti alla posizione del video le cui coordinate X e Y sono contenute nei registri X e Y. (Notate che la numerazione delle righe e delle colonne parte da zero). Affinché TV.PTR punti alla terza colonna da sinistra e alla quinta riga dall'alto occorre includere nel programma principale le seguenti linee:

LDX #2	La prima colonna è la zero, la seconda è la uno, la terza la due, ecc.
LDY #4	La prima riga è la zero, la seconda è la uno e così via
JSR TVTOXY	Setta TV.PTR alla posizione di coordinate X e Y contenute nei registri X e Y

Con quale logica funzionerà la subroutine TVTOXY? Potremmo memorizzare una tabella dove cercare l'indirizzo desiderato. Un computer può svolgere questo lavoro molto velocemente, ma la velocità non vale il sacrificio della quantità di memoria richiesta da una simile procedura. Se non ci interessa aspettare un po' di tempo in più, possiamo fare in modo che TVTOXY calcoli il valore da passare a TV.PTR. Come è possibile calcolare l'indirizzo di una data posizione X,Y sullo schermo? Sarà necessario conoscere le seguenti informazioni:

HOME	L'indirizzo in memoria del punto che si trova nella riga zero e colonna zero (cioè l'indirizzo più basso della memoria video)
RIGINC	Incremento di riga: la differenza degli indirizzi tra una riga e la successiva

Conoscendo i valori di HOME e RIGINC è possibile calcolare l'indirizzo di qualsiasi posizione X,Y:

HOME	indirizzo della posizione 0,0
+registro X	+coordinata X
+(registro Y)*RIGINC	+(coordinata Y)*RIGINC
TV.PTR	indirizzo della posizione X,Y

Provate a calcolare l'indirizzo di qualche posizione schermo e comparate il risultato con quello fornito dalla tabella della memoria video del vostro computer (ricordatevi che contiamo le righe e le colonne partendo da zero, non da uno).
A questo punto se la subroutine TVTOXY esegue questi calcoli, non sarà più necessario cercare gli indirizzi nella tabella e potremo scrivere il programma per il video in termini di coordinate cartesiane.
Eseguendo solamente questi calcoli, però, la routine non è perfettamente funzionale; infatti, cosa succederebbe se il contenuto del registro Y fosse maggiore del numero di righe visualizzabili? TV.PTR punterebbe ad una locazione esterna all'area di memoria video e TVTOXY memorizzerebbe dei caratteri in aree di memoria non previste o già occupate probabilmente dal programma principale, causando non pochi problemi.
Spesso la ricerca degli errori nei programmi è complicata; faremo così in

modo che la subroutine TVTOXY, oltre che calcolare gli indirizzi del video, controlli anche l'ammissibilità dei valori contenuti nei registri X e Y. Per poter compiere questi controlli ci occorrono ulteriori dati:

TVCOL	Numero di colonne del video contando da zero
TVRIG	Numero di righe del video contando da zero

Riassumendo, quindi, ci occorreranno quattro dati:

HOME
RIGINC
TVRIG
TVCOL

Se memorizziamo questi dati in un blocco di memoria, TVTOXY dovrà semplicemente consultare questa piccola tabella per sapere tutto ciò che gli occorre.
TVTOXY sarà quindi implementato come segue:

TVTOXY

```
TVTOXY  SEC             X fuori dai limiti?
        CPX TVCOL
        BCC X.OK        No, va bene così
        LDX TVCOL       Sì, setta X al massimo valore con-
                        sentito

X.OK    SEC             Y fuori dai limiti?
        CPY TVRIG
        BCC Y.OK        No, va bene così
        LDY TVRIG       Sì, setta Y al massimo valore con-
                        sentito

Y.OK    LDA HOME        Setta TV.PTR=HOME
        STA TV.PTR
        LDA HOME+1
        STA TV.PTR+1

        TXA             Somma X a TV.PTR
        CLC
        ADC TV.PTR
        BCC SETCOL
        INC TV.PTR+1
        CLC
```

```
SETCOL   CPY #0           Somma TV.PTR a (Y*RIGINC)
         BEQ FINE
LOOP     CLC
         ADC RIGINC
         BCC NEXT
         INC TV.PTR+1
NEXT     DEY
         BNE LOOP
FINE     STA TV.PTR
         RTS              Ritorna al programma principale
```

TVGIU-TVVAI-TVPIU

Usando la subroutine TVTOXY siamo in grado di far sì che il puntatore TV.PTR indirizzi una locazione della memoria video corrispondente alle coordinate X e Y desiderate. Sarebbe perciò anche molto utile riuscire a modificare TV.PTR in relazione al valore corrente. Per esempio, dopo aver visualizzato un carattere sullo schermo, potremmo voler indirizzare la successiva posizione a destra o quella immediatamente sotto, oppure perfino saltare all'ennesima posizione da dove ci troviamo. Per tutti questi casi faranno comodo le subroutine TVGIU, TVVAI E TVPIU:

TVGIU, TVVAI, TVPIU

```
TVGIU    LDA RIGINC       Sposta TV.PTR alla riga successiva
         CLC
         BCC TVPIU        Salto incondizionato

TVVAI    LDA #1           Salta una locazione video incre-
                          mentando il puntatore TV.PTR

TVPIU    CLC              Somma il contenuto dell'accumula-
         ADC TV.PTR       tore ai due byte in pagina zero ri-
         BCC NEXT         servati a TV.PTR
         INC TV.PTR+1

NEXT     STA TV.PTR
         RTS              Torna al programma principale
```

Notate che TVGIU e TVVAI fanno uso della routine TVPIU che richiede l'accumulatore settato con il numero di locazioni da saltare. Per TVGIU e

TVVAI l'accumulatore invece conterrà rispettivamente RIGINC e 1.
Potrebbe sembrare che TVPIU non sia lunga abbastanza per costituire una subroutine. Ogni programma che chiama TVPIU potrebbe eseguire la somma senza saltare ad una subroutine e questo ci farebbe risparmiare alcuni cicli macchina al prezzo di pochi byte. Però, in futuro potremo rendere TVPIU molto più sofisticata ed è questo il motivo che ci induce a creare una subroutine.
Potremo migliorare TVPIU in modo da compiere un controllo sui valori di TV.PTR per evitare un indirizzamento al di fuori della memoria video. Questo controllo sarebbe molto macchinoso se compiuto dal programma chiamante, ma se lo inseriamo all'interno di TVPIU ogni programma che chiama questa subroutine godrà dei benefici della modifica.

VEDCAR

Con TV.SCR siamo in grado di visualizzare un carattere ASCII nella corrente posizione di schermo, e con TVVAI possiamo spostarci alla posizione successiva. Perché non fondere le due routine e creare una subroutine che visualizzi l'immagine di un dato carattere ASCII nella corrente posizione sullo schermo e incrementi automaticamente il puntatore TV.PTR in modo che punti alla successiva posizione? Questo renderebbe molto facile la visualizzazione di una stringa di caratteri in posizioni di schermo successive. Poiché questa subroutine permetterà all'utilizzatore di vedere un carattere sarà chiamata VEDCAR:

```
VEDCAR    JSR TV.SCR     Visualizza nella corrente posizione
                         il carattere corrispondente al codice
          JSR TVVAI      Avanza fino alla posizione successiva
          RTS
```

Possiamo persino inserire VEDCAR tra le linee di programma viste per TVGIU, TVVAI e TVPIU aggiungendo una nuova linea di programma immediatamente prima di TVVAI. (Vedi in Appendice C1, il listato Assembler di queste utility che include anche alcuni controlli degli errori all'interno di TVPIU).

VEDBYT

Con le utility presentate in precedenza siamo in grado di visualizzare un carattere ASCII nella posizione corrente ma non abbiamo utility che visualizzano un byte in esadecimale. Vediamo di crearne una.

Chiameremo questa utility VEDBYT poiché ci permetterà di vedere un byte.
Per visualizzare un byte in notazione esadecimale, il programma principale, prima di chiamare VEDBYT, dovrà compiere tre semplici operazioni:

1. settare il puntatore TV.PTR in modo che punti alla posizione desiderata;
2. caricare l'accumulatore con il byte da visualizzare;
3. chiamare VEDBYT.

La Figura 5.1 rappresenta il diagramma di flusso di VEDBYT.

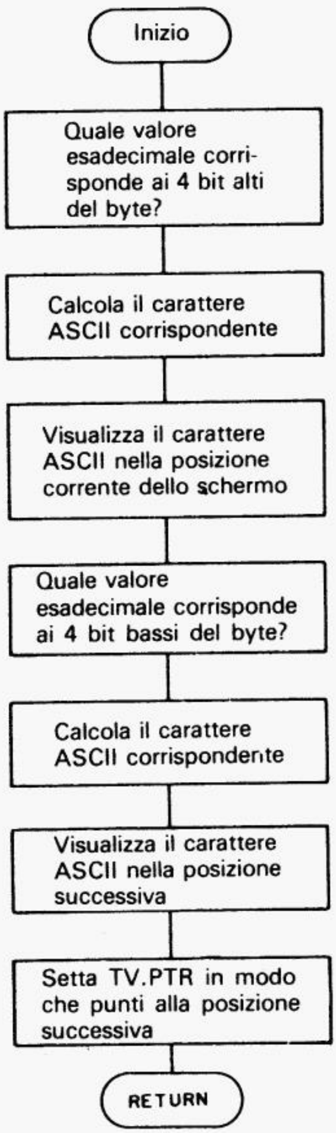


Figura 5.1 Diagramma di flusso della routine VEDBYT che permette di visualizzare un byte in esadecimale

VEDBYT permetterà di visualizzare un dato byte come due caratteri ASCII nella corrente posizione dello schermo e all'uscita della subroutine TV.PTR punterà alla successiva posizione dello schermo.
Il compito di VEDBYT è quello di determinare il carattere ASCII corrispondente al valore esadecimale dei 4 bit alti (MSB), memorizzare quel carattere ASCII nella locazione indicata da TV.PTR e poi visualizzare il carattere ASCII corrispondente al valore dei 4 bit bassi (LSB) nella posizione successiva.
Per svolgere questo compito VEDBYT richiede una subroutine che converta il valore esadecimale in carattere ASCII. Chiameremo quest'altra routine, ASCII.
La routine ASCII eseguirà la traduzione del valore esadecimale contenuto nell'LSB dell'accumulatore da esadecimale ad ASCII, ignorando l'MSB.
Supponendo che questa routine esista potremo quindi scrivere il programma VEDBYT:

VEDBYT

VEDBYT	PHA	Salva l'accumulatore
	LSR A LSR A LSR A LSR A	Trasla l'MSB nella posizione occupata dall'LSB
	JSR ASCII	Determina il carattere corrispondente all'LSB dell'accumulatore
	JSR VEDCAR	Visualizza il carattere ASCII nella posizione corrente e avanza di una posizione
	PLA	Riprende il valore originale dell'accumulatore
	JSR ASCII	Determina il carattere ASCII corrispondente all'LSB
	JSR VEDCAR	Visualizza il carattere ASCII nella posizione successiva e avanza di una posizione
	RTS	Ritorna al programma principale

La routine ASCII al momento non esiste ancora, quindi, per rendere completo VEDBYT la scriveremo così:

ASCII

ASCII	AND #$0F	Azzera l'MSB dell'accumulatore
	CMP #$0A BMI DECIML	L'accumulatore è maggiore di 9?
	ADC #6	Se sì, deve essere compreso tra A e F e bisogna sommare $36 per ottenere il carattere ASCII corrispondente. (Sommeremo prima $6 e poi $30)
DECIML	ADC #$30	Se no, è compreso tra 0 e 9. Sommando $30 otterremo l'ASCII corrispondente
	RTS	Ritorna al programma principale

TVHOME, CENTRO

Siamo a questo punto in grado di visualizzare un carattere ASCII o un byte nella corrente posizione dello schermo, e siamo inoltre in grado di settare la posizione dello schermo in ogni punto X e Y desiderato. Sarebbe utile poter settare TV.PTR a certe posizioni di uso comune; posizioni che potrebbero essere necessarie a più di un programma chiamante. Per esempio, queste posizioni potrebbero essere HOME (0,0) o il centro dello schermo:

TVHOME, CENTRO

TVHOME	LDX #0	Setta TV.PTR alla colonna più a sinistra della prima riga dello schermo
	LDY #0 JSR TVTOXY	
	RTS	Ritorna al programma principale

```
CENTRO   LDA TVRIG      Carica in A il numero totale delle
                        righe
         LSR A          Divide per due
         TAY            Y contiene ora il numero della riga
                        centrale

         LDA TVCOL      Carica in' A il numero totale delle
                        colonne
         LSR A          Divide per due
         TAX            X contiene ora il numero della co-
                        lonna centrale

         JSR TVTOXY     Setta TV.PTR alle coordinate X e Y
         RTS            Ritorna al programma principale
```

TVSALV E TVREC

Le utility presentate qui di seguito ci permetteranno di salvare l'attuale posizione sullo schermo e di recuperarla all'occorrenza. Saremo in grado di fare questo inserendo e recuperando dallo stack il puntatore TV.PTR.

TVSALV

```
TVSALV   PLA            Legge l'indirizzo di ritorno dallo
                        stack

         TAX            Lo salva in X
         PLA
         TAY            e in Y

         LDA TV.PTR+1   Legge TV.PTR e lo salva nello stack
         PHA
         LDA TV.PTR
         PHA

         TYA            Rimette nello stack
         PHA            l'indirizzo di ritorno
         TXA
         PHA

         RTS            Ritorna al programma principale
```

TVREC

```
TVREC     PLA              Legge l'indirizzo di ritorno dallo
                           stack
          TAX              Lo salva in X
          PLA
          TAY              e in Y

          PLA              Recupera TV.PTR dallo stack
          STA TV.PTR
          PLA
          STA TV.PTR+1

          TYA              Rimette nello stack
          PHA              l'indirizzo di ritorno
          TXA
          PHA
          RTS              Ritorna al programma principale
```

Il programma principale è adesso in grado di salvare la posizione attuale sullo schermo con una semplice linea di codice — JSR TVSALV —, modificare a piacimento il puntatore TV.PTR e poi ristabilire il valore iniziale con JSR TVREC.

Pulire lo schermo

Adesso che siamo in grado di visualizzare un carattere o un byte in ogni posizione dello schermo, proviamo a scrivere una subroutine che ci permetta di pulire parte o tutto lo schermo. Una subroutine, che chiameremo PULTV cancellerà tutto il contenuto dello schermo, mentre un'altra subroutine, PULXY, cancellerà il rettangolo le cui dimensioni x e y sono contenute nei registri X e Y. Perciò il programma principale potrà richiamare PULTV per cancellare tutto lo schermo, oppure PULXY per cancellare una determinata area lasciando il resto inalterato.
Per eseguire queste operazioni, sarà necessario fare in modo che il puntatore TV.PTR punti al vertice superiore sinistro del rettangolo da cancellare e che i registri X e Y contengano le dimensioni (rispettivamente larghezza e altezza) dell'area da cancellare.

```
PULTV     JSR TVSALV       Salva i byte di pagina zero che sa-
                           ranno cambiati
          JSR TVHOME       Setta il puntatore alla posizione 0,0
```

```
        LDX TVCOL       Carica in X e Y le
        LDY TVRIG       dimensioni dello schermo
        JSR PULXY       Cancella X colonne e Y righe dalla
                        posizione attuale
        JSR TVREC       Resetta i byte di pagina zero
        RTS             Ritorna con lo schermo pulito e la
                        pagina zero inalterata

PULXY   STX COL         Setta il numero di colonne da can-
                        cellare
        TYA
        TAX             Setta il numero di righe da cancel-
                        lare
PULRIG  LDA NULL        Carica in A il valore corrispondente
                        a uno spazio bianco
        LDY COL         Carica in Y il numero di colonne
                        da cancellare
PULPOS  STA (TV.PTR),Y  Cancella la posizione scrivendo uno
                        spazio bianco
        DEY             Decrementa Y per la posizione suc-
                        cessiva
        BPL PULPOS      Se non finito cancella la .posizione
                        successiva
        JSR TVGIU       Se la riga è finita passa alla succes-
                        siva
        DEX             È finita l'ultima riga?
        BPL PULRIG      Se no, cancella la successiva riga
        RTS             Ritorna al programma principale
COL     .BYTE 0         Variabile: contiene il numero di co-
                        lonne da cancellare
```

Esistono molte altre utility che possono essere sviluppate, ma le routine presentate in questo capitolo vogliono essere solo una base di partenza. Adesso un programma si potrà avvalere delle seguenti subroutine per ottenere i risultati descritti:

ASCII	Converte in ASCII i 4 LSB contenuti in A
CENTRO	Setta la posizione corrente al centro dello schermo
PULTV	Cancella il contenuto dello schermo
PULXY	Cancella il rettangolo di dimensioni X e Y
TVGIU	Sposta la posizione di una riga verso il basso
TVHOME	Sposta la posizione corrente nell'origine
TVPIU	Somma a TV.PTR il contenuto di A

TVREC	Recupera dallo stack la posizione precedentemente salvata
TVSALV	Salva nello stack l'attuale posizione
TV.SCR	Visualizza il carattere ASCII contenuto in A
TVVAI	Avanza alla successiva posizione
TVTOXY	Setta la corrente posizione alle coordinate X e Y
VEDBYT	Visualizza il contenuto di A in sistema esadecimale e avanza di una posizione
VEDCAR	Visualizza il contenuto di A come carattere ASCII e avanza di una posizione

Per mezzo di queste utility il programma principale potrà gestire lo schermo senza preoccuparsi della memoria video e dei puntatori in pagina zero. Il programma dovrà solamente gestire la posizione corrente sullo schermo, senza preoccuparsi dell'indirizzamento della memoria video.

Capitolo

Il Monitor Visibile 6

Assemblaggio manuale

Un assemblatore è un potente strumento di programmazione, ma cosa si può fare se non è disponibile? È possibile scrivere programmi per il 6502 senza un assemblatore?
Non solo è possibile assemblare manualmente un programma, ma tutto il software presentato in questo libro è stato, in origine, assemblato e inserito nel computer manualmente.
L'assemblaggio manuale impone una certa disciplina al programmatore. Infatti gli indirizzi di salto vanno calcolati contando avanti e indietro in esadecimale, il che non è molto semplice. I programmi presentati sono scritti come una serie di subroutine annidate e provate individualmente; questa è una buona norma anche qualora si usi un assemblatore, ma diventa indispensabile quando si è costretti all'assemblaggio manuale.
Una volta scritto un programma in codice macchina, come potremo inserirlo in memoria? Sarà possibile leggerlo sulla carta, ma come lo presenteremo al 6502?
Un programma chiamato *monitor* ci permette di esaminare e modificare il contenuto della memoria. Permette inoltre di eseguire un programma memorizzato. Il C-64 e il VIC-20 non hanno un monitor incorporato; sono disponibili monitor molto sofisticati su disco o su cassetta, ma dal punto di vista dell'apprendimento della teoria non è strettamente necessario possederne uno. Così, prima di correre a comperare un monitor, proviamo a esaminarne uno molto semplice.
Quello che prenderemo in esame è usato sull'OSI C I-P (*Ohio Scientific*

Challenger I-P). Viene presentato qui solo come esempio poiché non è disponibile per i computer Commodore.

Un semplice monitor di codice macchina

È possibile richiamare il monitor dell'OSI semplicemente premendo il tasto BREAK e il tasto M. Il monitor cancella lo schermo e si presenta come si può vedere in Figura 6.1.
La videata è costituita da due campi di caratteri esadecimali: un campo degli indirizzi e uno dei dati. Nella Figura 6.1 è possibile vedere che $A9 è l'attuale valore dell'indirizzo $0000. Il monitor dell'OSI può funzionare in due modi: indirizzi e dati. Quando si trova in modo indirizzi è possibile visualizzare il contenuto di ogni indirizzo semplicemente scrivendo l'indirizzo sulla tastiera. Ogni carattere esadecimale entrerà nel campo degli indirizzi da destra, quindi per impostare l'indirizzo $FE0D sarà sufficiente premere in sequenza i tasti F, E, 0 e D. Per poter cambiare il contenuto di un indirizzo bisogna passare in modo dati. Quando il monitor

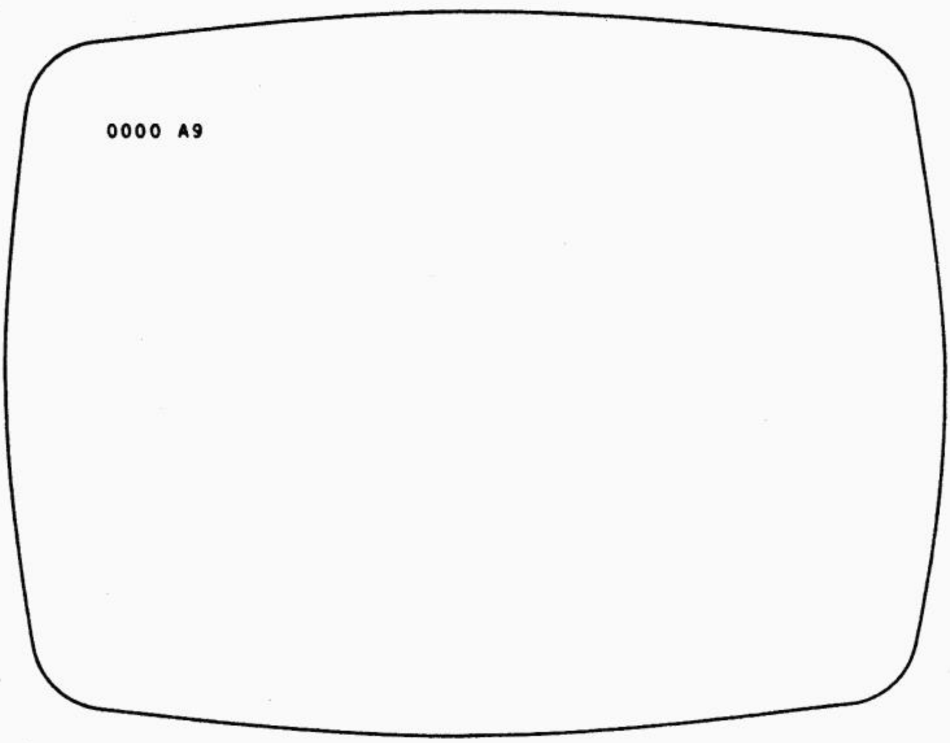


Figura 6.1 Il monitor dell'OSI

dell'OSI è in questo modo, i caratteri esadecimali digitati da tastiera si inseriranno nel campo dati e per avanzare di un indirizzo alla volta sarà sufficiente premere il tasto RETURN. Sfortunatamente questa possibilità non è disponibile nel modo indirizzi e in nessuno dei due modi è possibile retrocedere di una locazione in memoria.

Attenzione, però, il monitor dell'OSI potrebbe confondervi. Se il monitor è in modo dati e voi digitate sulla tastiera un carattere esadecimale, quel carattere verrà visualizzato nel campo dati sullo schermo. Si presume che quel carattere esadecimale venga anche memorizzato nella locazione visibile sullo schermo, ma questo potrebbe non essere vero; infatti, il monitor dell'OSI visualizza i dati che si vogliono memorizzare piuttosto che il reale contenuto dell'indirizzo. Per esempio, se si cerca di cambiare un indirizzo di ROM, il monitor confermerà che il dato desiderato è stato memorizzato, se però andiamo ad ispezionare quell'indirizzo (passando in modo indirizzi e digitando quell'indirizzo) sicuramente vedremo che non è stato cambiato nulla. Morale: non è possibile scrivere nelle memorie di sola lettura, ma il monitor dell'OSI vi fa credere il contrario.

Il monitor dell'OSI vi può confondere in altri modi. Per esempio, bisogna ricordarsi se si è in modo indirizzi o dati e, per passare da un modo all'altro, è necessaria una ben determinata sequenza di caratteri (bisognerà quindi ricordare anche questi codici). Inoltre, con questo monitor non è molto facile l'inserimento delle stringhe di caratteri ASCII. Per inserire un messaggio in ASCII sarà necessario consultare una tabella dei codici ASCII (vedi Appendice A2), ricavare il valore esadecimale di ogni carattere e inserirlo in questa forma. Inoltre, dovendo rileggere il messaggio sarà necessario il procedimento inverso.

Vale la pena di esaminare la maniera in cui il monitor dell'OSI esegue un programma memorizzato. Quando si preme il tasto G, il monitor esegue un salto incondizionato (JMP) all'indirizzo visualizzato, quindi trasferisce il controllo al programma selezionato. Questo procedimento però viene eseguito in modo tale che il programma selezionato deve terminare con un altro salto incondizionato per ritornare al monitor. Questo fatto costringe a scrivere programmi che terminano con l'istruzione JMP piuttosto che subroutine che finiscono di regola con l'istruzione RTS.

I programmi che terminano con JMP non sono facilmente usabili come base per altri programmi, mentre al contrario le subroutine possono essere incorporate in un software personalizzato. Sarebbe quindi meglio usare un monitor di sistema che permetta di saltare all'indirizzo visualizzato come ad una subroutine piuttosto che a un programma. Un monitor di questo tipo aiuterebbe ad acquisire un concreto stile di programmazione, introducendo l'utente alla programmazione strutturata. In questo capitolo svilupperemo un monitor di sistema con queste caratteristiche.

Obiettivi

Se vi è già capitato di usare un monitor di sistema avrete probabilmente pensato di apportargli una qualche miglioria. Basandoci sulla nostra esperienza un monitor dovrebbe essere:

1. *Accurato*: il campo dati deve visualizzare il reale contenuto dell'indirizzo e non quello che dovrebbe esserci.
2. *Comodo da usare*: deve essere possibile avanzare e retrocedere di un passo alla volta in memoria, qualsiasi sia il modo selezionato. Deve inoltre esserci la possibilità di inserire stringhe di caratteri ASCII direttamente da tastiera, senza dover consultare tabelle e di conseguenza deve essere possibile visualizzare queste stringhe in codice ASCII e non come coppie di valori esadecimali.
3. *Orientato alla programmazione strutturata*: deve essere possibile richiamare l'indirizzo visualizzato come una subroutine piuttosto che saltare all'indirizzo.
4. *Utile nella ricerca degli errori*: deve essere possibile caricare nei registri del 6502 valori definiti dall'utente prima di richiamare l'indirizzo visualizzato. In questo modo sarà possibile provare le subroutine con differenti valori e, finita l'esecuzione della subroutine, il monitor dovrà visualizzare i nuovi valori dei registri. In questo modo, essendo possibile esaminare le modificazioni che avvengono, sarà più facile stabilire se la routine funziona correttamente o meno.
   Proprio per questa possibilità di poter visualizzare il contenuto dei registri prima e dopo l'esecuzione di una subroutine, chiameremo questo monitor *Monitor Visibile*. La Figura 6.2 mostra una videata tipica del Monitor Visibile.

Il Monitor Visibile

Noterete che la videata di Figura 6.2 presenta ben sette campi e non due come succede con il monitor OSI. I primi due campi (campo 0 e 1) sono identici al monitor OSI, cioè rappresentano un indirizzo e il suo contenuto in esadecimale. Il campo 2 è una rappresentazione grafica o ASCII del valore contenuto nell'indirizzo. Se l'indirizzo visualizzato contiene un carattere ASCII, allora nel campo 2 ci sarà l'immagine del carattere, numero o segno di interpunzione corrispondente, altrimenti ci sarà l'immagine di un carattere grafico caratteristico del Commodore, ma non contemplato dal codice ASCII.
I campi da 3 a 6 rappresentano i quattro registri del 6502: A (l'accumulatore), X (il registro X), Y (il registro Y) e P (il registro di stato).

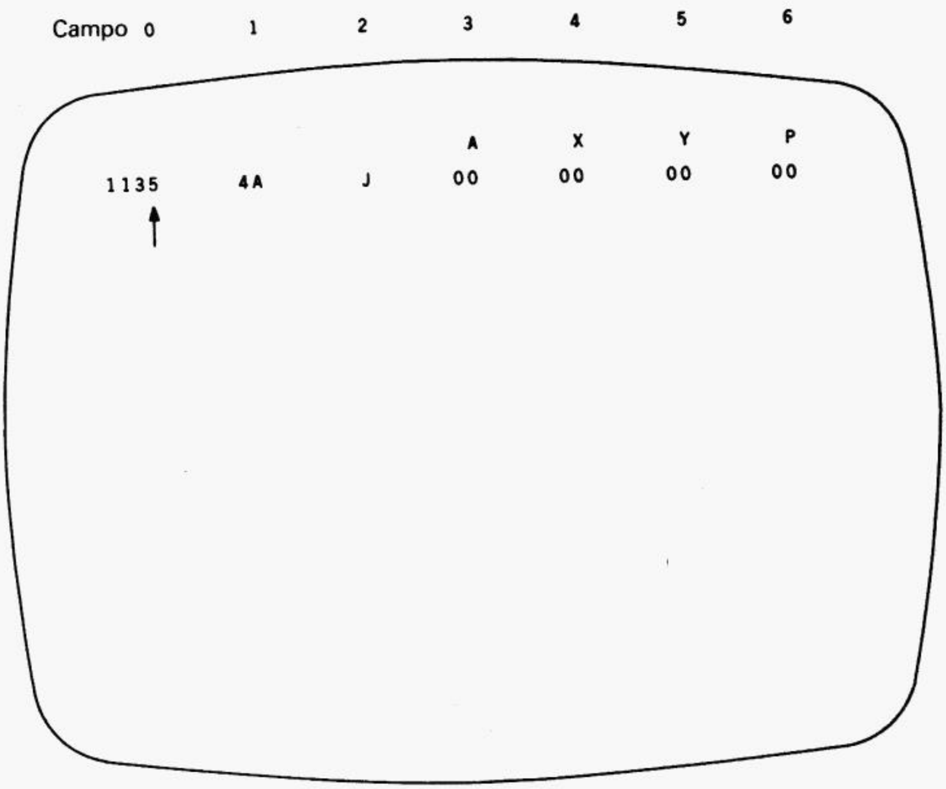


Figura 6.2 Come si presenta il Monitor Visibile con i campi numerati

Quando si premerà il tasto G per eseguire un programma, i registri verranno caricati con il valore visualizzato e, quando il controllo ritornerà al monitor, i registri saranno visualizzati di nuovo.
Oltre ai sette campi il nostro monitor visualizzerà anche una freccia indicante uno dei sette campi. Per modificare uno dei campi sarà necessario fare in modo che la freccia indichi il campo da modificare. Premendo il tasto di movimento cursore sarà possibile cambiare la posizione della freccia e puntare così il campo desiderato. Attenzione: il tasto di movimento cursore è unico; premendolo senza lo SHIFT ci si potrà spostare a destra, mentre premendolo con lo SHIFT ci si sposterà a sinistra.
Abbiamo scelto la barra spaziatrice per incrementare l'indirizzo e il tasto RETURN per decrementarlo: ovviamente è possibile usare una qualsiasi coppia di tasti come, ad esempio, + e −.
Con queste possibilità di visualizzazione e questi tasti funzione dovrebbe essere possibile costruire un monitor potente e maneggevole.

Dati

Prima di sviluppare la struttura e il programma del Monitor Visibile è utile fare il punto sulle variabili e sui puntatori che occorreranno.
Il Monitor Visibile deve conoscere in qualche modo l'indirizzo da visualizzare nel campo 0. Questo è possibile per mezzo di un puntatore che specifichi l'indirizzo da visualizzare. Chiameremo questo puntatore SELEZ poiché punterà all'indirizzo selezionato. Quindi, ad ogni pressione della barra spaziatrice, SELEZ dovrà essere incrementato, mentre il tasto RETURN dovrà decrementarlo rendendo così possibile lo spostamento avanti e indietro in memoria.
L'utente dovrà inoltre essere in grado di cambiare il contenuto dei registri. Poiché ci sono quattro registri, sarà necessario riservare quattro byte in memoria, uno per ogni registro. Se riserviamo questi byte in locazioni contigue sarà utile riferirci a questo blocco chiamandolo REGISTRI o meglio REGI (REGISTRI sarebbe un'etichetta troppo lunga per essere accettata dall'Assembler che considera valide etichette lunghe non più di sei caratteri). Sarà infine necessario specificare il campo selezionato. Siccome verrà selezionato un solo campo per volta, faremo uso di una variabile, CAMPO, il cui valore specificherà il campo selezionato. Quindi, quando si dovrà passare al campo successivo sarà sufficiente incrementare questa variabile e, per passare al campo precedente, sarà sufficiente decrementarla. Se il valore della variabile CAMPO non sarà compreso nei limiti stabiliti (cioè esterno all'intervallo 0-6) il monitor dovrà assegnarle un valore appropriato.
Le linee di programma sorgente riportate di seguito dichiarano queste variabili con la sintassi accettata dall'Assembler dell'OSI:

Variabili

```
SELEZ    .WORD 0       Puntatore di indirizzo
REG.A    .BYTE 0       Valore dell'accumulatore
REG.X    .BYTE 0       Valore del registro X
REG.Y    .BYTE 0       Valore del registro Y
REG.P    .BYTE 0       Valore del registro P
CAMPO    .BYTE 0       Numero del campo selezionato
         REGI=REG.A
```

Struttura

Si è cercato di mantenere il Monitor Visibile altamente modulare per facilitare ulteriori modifiche o ampliamenti. È stata comunque tenuta ad esempio la struttura di Figura 6.3. Come si può chiaramente intendere, il Monitor Visibile è un loop. Questo loop consiste nel visualizzare sullo schermo i campi ed eseguire le modifiche richieste dall'utente con i tasti funzione.

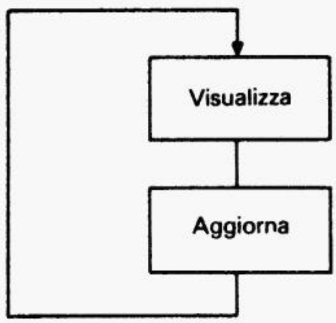


Figura 6.3 Una semplice struttura per programmi interattivi di visualizzazione

Con questo diagramma in mente possiamo cominciare a scrivere le linee che costituiscono il primo livello del Monitor Visibile (VISMON):

VISMON

```
VISMON   PHP          Salva gli indicatori del registro P
LOOP     JSR VISUAL   Visualizza i campi
         JSR AGGIOR   Esegue le operazioni richieste
         CLC
         BCC LOOP     Ricomincia il ciclo
```

Questo è solo il primo abbozzo del Monitor Visibile e ovviamente non funzionerà senza le subroutine VISUAL e AGGIOR. Potrebbe sembrare sconveniente scrivere una subroutine che ne richiama altre due, ma strutturando il monitor in questa maniera sarà più facile svilupparlo, modificarlo e correggere gli eventuali errori.
Inizieremo con lo studio del modulo VISUAL perché il video costituisce l'interfaccia tra macchina e utente; questo modulo dovrà rispettare le nostre esigenze. Una volta possibile la visualizzazione delle informazioni, potremo scrivere la routine AGGIOR.

Visualizzazione

La Figura 6.2 mostra la videata che vogliamo ottenere. Come potete notare, questa videata è composta da tre linee di caratteri: la linea dei titoli, quella dei dati e quella che contiene la freccia di selezione. La prima linea, quella dei titoli, identifica i quattro campi riguardanti i registri con le lettere A, X, Y, P. La linea dei dati visualizza un indirizzo, il suo contenuto (in forma esadecimale e come carattere) e il valore contenuto nei registri del 6502. La terza linea contiene una freccia che punta a uno dei campi visualizzati. Poiché a questo punto abbiamo definito quello che ci occorre in termini di linea dei titoli, linea dei dati e linea indice, possiamo disegnare il diagramma a blocchi della routine VISUAL come mostrato in Figura 6.4.
Tenendo presente questo diagramma scriviamo il primo abbozzo della subroutine VISUAL.



Figura 6.4 Routine per visualizzare le informazioni richieste

VISUAL

VISUAL	JSR PULMON	Cancella la parte del video riservata al monitor
	JSR LINE.1	Visualizza la linea dei titoli
	JSR LINE.2	Visualizza la linea dei dati

```
          JSR LINE.3       Visualizza la freccia indice
          RTS              Ritorna al programma principale
```

Come noterete, adesso invece di una subroutine dovremo scriverne quattro: PULMON, LINE.1, 2 e 3. Man mano però che il numero delle subroutine aumenta, diminuisce la loro complessità.
Prima di visualizzare qualche cosa sarà utile pulire quella parte dello schermo usata dal monitor.
Siccome abbiamo già creato un'utility che permette di cancellare X colonne e Y righe, sarà sufficiente settare TV.PTR nella posizione di home (0,0), caricare i registri X e Y con i valori appropriati ed eseguire un salto alla subroutine PULXY.

```
PULMON    LDX #2           Setta TV.PTR
          LDY #2           alla riga 2 e colonna 2
          JSR TVTOXY
          LDX TVCOL        Cancella per tutta la larghezza
          LDY #3           Cancella le prime 3 righe
          JSR PULXY        Esegue la cancellazione
          RTS              Ritorna al programma principale
```

Linea dei titoli

La subroutine LINE.1 ha il compito di visualizzare la linea dei titoli. Memorizzeremo in qualche locazione di memoria la stringa di caratteri "A X Y P" e ci riferiremo a queste locazioni con il nome di TITOLI. Sarà quindi sufficiente che la subroutine LINE.1 legga 10 byte dal blocco TITOLI e li riscriva nell'appropriata posizione dello schermo. Con questo procedimento saremo in grado di visualizzare la linea dei titoli:

LINE.1

```
LINE.1    LDX #11          Coordinata X del titolo "A"
          LDY #0           Coordinata Y del titolo "A"
          JSR TVTOXY       Setta TV.PTR alle coordinate X e Y
          LDY #0           Visualizza
          STY TITCOL       Inizializza il contatore delle colonne
TITLOP    LDA TITOLI,Y     Legge un carattere
          JSR VEDCAR       Lo visualizza
          INC TITCOL       Si prepara per il successivo
```

```
          LDY TITCOL      Usa TITCOL come indice
          CPY #10         È l'ultimo carattere?
          BNE TITLOP      No, prosegue con il successivo
          RTS             Ritorna al programma principale
TITOLI    .BYTE 'A X'     Questi sono i caratteri
          .BYTE 'Y P'     da visualizzare
TITCOL    .BYTE 0         Questo è il contatore
```

Linea dei dati

Le operazioni da compiere per visualizzare la linea dei dati saranno un po' più complicate per due motivi. In primo luogo i dati da visualizzare cambieranno di volta in volta e, secondariamente, la maggior parte dei campi conterrà dati in forma esadecimale. La visualizzazione di un byte in forma esadecimale richiede più lavoro che non visualizzare un carattere ASCII.
Fortunatamente abbiamo un'utility, VEDBYT, che ci permetterà di compiere questa operazione. Possiamo quindi scrivere la routine che visualizza la seconda linea:

LINE.2

```
LINE. 2   LDX #0          Coordinata X dell'inizio linea
          LDY #1          Coordinata Y della linea dati
          JSR TVTOXY      Setta TV.PTR
          LDA SELEZ+1     Visualizza il nybble alto
          JSR VEDBYT
          LDA SELEZ       Visualizza quello basso
          JSR VEDBYT
          JSR TVVAI       Si sposta di una posizione verso de-
                          stra
          JSR LEGBYT      Legge il valore del byte selezionato
          PHA             Lo salva
          JSR VEDBYT      Lo visualizza
          JSR TVVAI
          PLA             Recupera il valore
          JSR VEDCAR      Lo visualizza in codice ASCII
          JSR TVVAI       Visualizza i registri
          LDX #0
VEDREG    LDA REGI,X      Legge i registri
          JSR VEDBYT      Visualizza il contenuto
```

```
        JSR TVVAI
        INX              Si prepara per il successivo
        CPX #4           Visualizzati tutti e 4?
        BNE VEDREG       No, legge il successivo
        RTS              Ritorna al programma principale
```

Noterete che la subroutine LINE.2 non conosce il valore del byte selezionato. Questo valore viene letto dalla subroutine LEGBYT che legge il contenuto dell'indirizzo puntato da SELEZ. Se SELEZ è un puntatore in pagina zero, LEGBYT, sarà una routine molto semplice e potrà avvalersi dei vantaggi dell'indirizzamento indiretto

```
LEGBYT  LDY #0           Legge il byte a cui
        LDA (SELEZ),Y    punta SELEZ
        RTS
```

Però SELEZ non è un puntatore in pagina zero; si trova infatti in pagina $32. Come sappiamo il 6502 non possiede un indirizzamento che permetta di caricare un registro per mezzo di un puntatore estraneo alla pagina zero.
Come sarà possibile leggere il valore contenuto nella locazione specificata da SELEZ? Ci riusciremo in due passaggi. Useremo un puntatore in pagina zero che sia uguale in valore a SELEZ e che quindi punterà allo stesso indirizzo e poi caricheremo l'accumulatore per mezzo di questo nuovo puntatore in pagina zero che chiameremo LEGPTR. Usando questa strategia la nostra routine si presenterà così:

```
LEGBYT  LDA SELEZ        Setta LEGPTR uguale a SELEZ
        STA LEGPTR
        LDA SELEZ+1
        STA LEGPTR+1
        LDY #0           Legge il byte specificato da
        LDA (LEGPTR),Y   LEGPTR
        RTS              Ritorna al programma (A=conte-
                         nuto dell'indirizzo puntato da
                         SELEZ)
```

Questo secondo tentativo caricherà l'accumulatore col valore della locazione specificata da SELEZ anche se questo puntatore non è in pagina zero. Attenzione però che settando LEGPTR uguale a SELEZ, la subroutine LEGBYT cambia il valore di LEGBYT, e questo può essere molto pericoloso. Cosa succederebbe per esempio se un altro programma facesse uso di LEGPTR? L'altro programma sarebbe "sabotato" da LEGBYT. Se permettiamo che la routine LEGBYT cambi il valore del puntatore LEGPTR,

dovremo essere sicuri che nessun altro programma usi LEGPTR. Questo risultato non è ottenibile se si vuole mantenere la coabitazione con le routine di I/O del vostro sistema. Quando si devono usare dei byte in pagina zero, come nel nostro caso, è necessario, dopo l'uso, risistemare i valori originali.
Quindi, la nostra routine LEGBYT sarà suddivisa in 4 parti: la prima salverà LEGPTR, la seconda setterà LEGPTR uguale a SELEZ, la terza caricherà l'accumulatore con il contenuto della locazione specificata e finalmente la quarta ripristinerà il valore originario di LEGPTR.
Questa versione di LEGBYT sarà certamente più ingombrante e più lenta, ma largamente più sicura:

```
LEGBYT    LDA LEGPTR
          PHA               Salva LEGPTR
          LDX LEGPTR+1

          LDA SELEZ
          STA LEGPTR        LEGPTR=SELEZ
          LDA SELEZ+1
          STA LEGPTR+1

          LDY #0            Legge il valore puntato da SELEZ
          LDA (LEGPTR),Y
          TAY

          PLA               Resetta LEGPTR
          STA LEGPTR
          STX LEGPTR+1
          TYA
          RTS               Ritorna al programma principale
```

La freccia indice

Questa routine visualizza la freccia indice che evidenzia il campo corrente:

LINE.3

```
LINE.3    LDY    CAMPO      Punta il campo corrente
          SEC               Se fuori limite
          CPY #7            punta il campo di default
          BCC CAM.OK
          LDY #0
          STY CAMPO
```

```
CAM.OK    LDA   CAMPI,Y    Legge a che colonna è il campo
                           corrente
          TAX              Questa sarà la coordinata X
          LDY #2           Coordinata Y
          JSR   TVTOXY     Setta TV.PTR

          LDA   FRECC      Visualizza la freccia
          JSR   VEDCAR     in quella posizione
          RTS
CAMPI     .BYTE 3,6,8      Colonne dove potrebbe
          .BYTE $0B,$0E    apparire la freccia
          .BYTE $11,$14
```

Ora che abbiamo tutte le routine occorrenti per visualizzare, vediamo come vengono legate assieme per formare una struttura.
Di seguito è visibile la gerarchia delle subroutine richiamate da VISUAL:

```
VISUAL
      LINE.1
      LINE.2
         LEGBYT
         VEDBYT
            ASCII
            TVPIU
         TVVAI
      LINE.3
```

Quando VISUAL sarà eseguito, verranno cancellate le prime quattro righe dello schermo e verranno visualizzate le tre linee ora descritte. Anche se il programma può sembrare complicato e lento, verrà eseguito in un batter d'occhio.

La sezione di aggiornamento

La routine AGGIOR permette di compiere determinate operazioni a seconda del tasto premuto. I tasti funzione che ci interessano sono:

Tasto	**Funzione**
→	Spostamento dell'indice verso destra
←	Spostamento dell'indice verso sinistra
Barra spaziatrice	Incremento dell'indirizzo visualizzato
RETURN	Decremento dell'indirizzo visualizzato

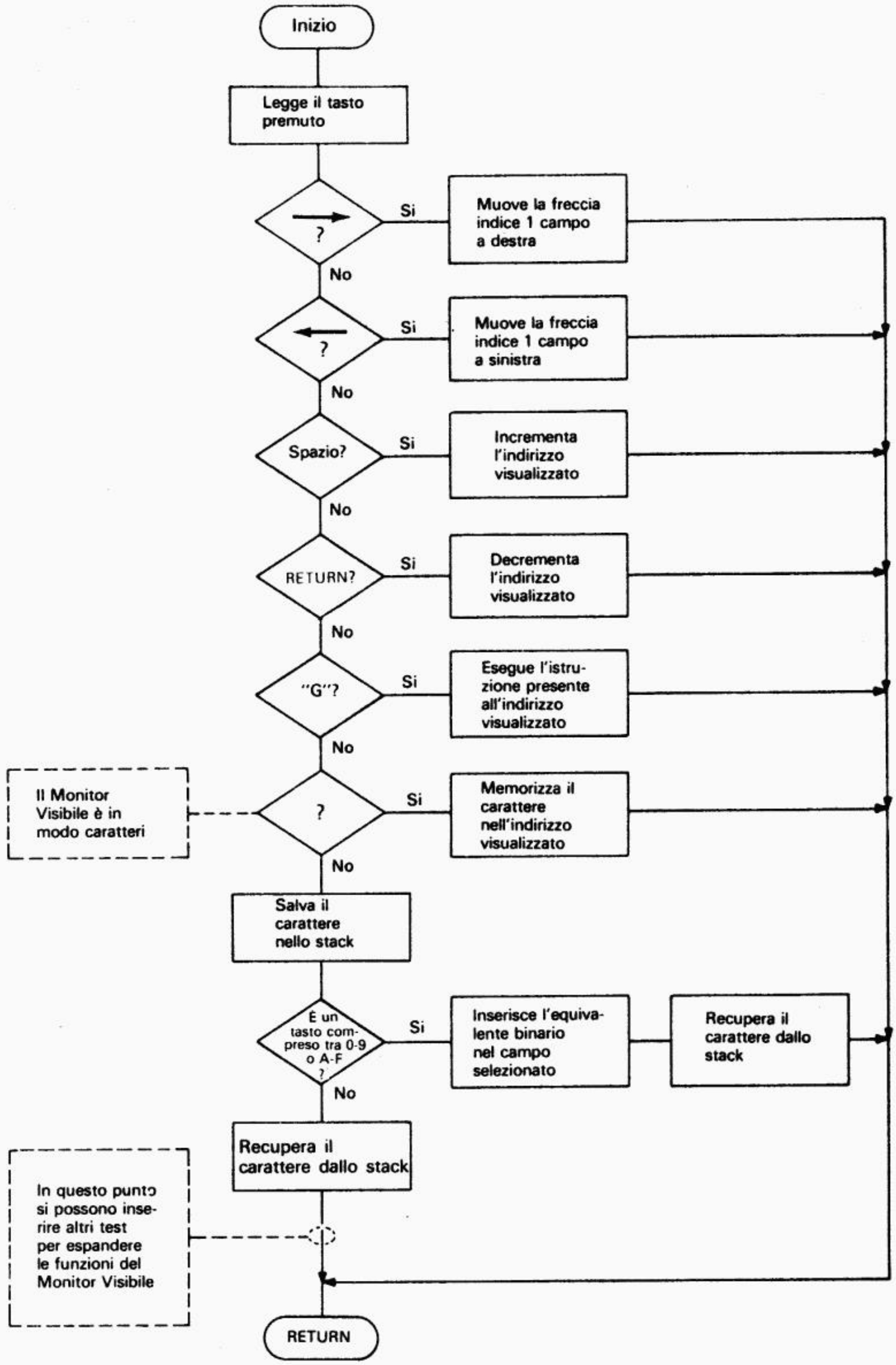


Figura 6.5 Diagramma di flusso della routine AGGIOR

Se l'indice è nei campi 1, 3, 4, 5 o 6:

da 0 a 9, da A a F	Inserimento del carattere esadecimale nel campo indicato dalla freccia

Se l'indice è nel campo 2:

qualsiasi	Inserimento del carattere ASCII corrispondente

Poiché il circuito video non necessita di *refresh* (riscrittura entro un certo tempo), la routine AGGIOR potrà funzionare senza problemi di temporizzazione. La routine AGGIOR può quindi aspettare indefinitamente un comando da tastiera e compiere le operazioni richieste.
Il diagramma di flusso di questa routine è visibile in Figura 6.5. Sarà possibile implementare ulteriori funzioni semplicemente aggiungendo test per uno o più caratteri e scrivendo le eventuali subroutine di ampliamento.

L'acquisizione del carattere

Per prima cosa dobbiamo essere in grado di leggere un input da tastiera. Naturalmente nel vostro sistema esiste già una routine in ROM che svolge questo compito. Come si può vedere nelle Appendici B1 e B2, abbiamo memorizzato gli indirizzi di questa routine in un puntatore che chiameremo ROMKEY, situato alla locazione $3008. Una volta settato ROMKEY sarà possibile leggere il valore del tasto premuto chiamando la subroutine LEGKEY che trasferisce il controllo alla routine contenuta in ROM.

```
LEGKEY     JMP (ROMKEY)
```

Adesso che siamo in grado di leggere il valore del tasto premuto, siamo anche in grado di scrivere la routine AGGIOR:

AGGIOR

```
AGGIOR   JSR LEGKEY    Legge il tasto premuto
         CMP #$1D      È il tasto →?
         BNE IF.LFT    Se no, esegue il test successivo
NEXT.F   INC CAMPO     Se sì, seleziona il campo successivo
         LDA CAMPO     Se l'indice era al campo più a de-
                       stra,
```

```
            CMP #7           si posiziona sotto quello più a
            BNE VIA 1        sinistra
            LDA #0
            STA CAMPO
VIA 1       RTS              Ritorna al programma principale
IF.LFT      CMP #$9D         È il tasto ←?
            BNE IF.SP        Se no, esegue il test successivo
PREV.F      DEC CAMPO        Se sì, seleziona il campo
                             precedente
            BPL VIA 2        Se l'indice era al campo più a
            LDA #6           sinistra si posiziona sotto quello
            STA CAMPO        a destra
VIA 2       RTS              Ritorna al programma principale
IF.SP       CMP #SPAZIO      È uno spazio?
            BNE IF.CR        Se no, esegue il test successivo
INC.SL      INC SELEZ        Se sì, avanza in memoria
            BNE VIA 3        incrementando il puntatore che
            INC SELEZ+1      specifica l'indirizzo visualizzato
VIA 3       RTS              Ritorna al programma principale
IF.CR       CMP #CR          È un RETURN?
            BNE IFCAR        Se no, esegue il test successivo
DEC.SL      LDA SELEZ        Se sì, retrocede in memoria
            BNE NEXT.1       decrementando il puntatore che
            DEC SELEZ+1      specifica l'indirizzo visualizzato
NEXT.1      DEC SELEZ
            RTS              Ritorna al programma principale
IFCAR       LDX CAMPO        L'indice è sotto il campo 2?
            CPX #2
            BNE IF.GO        Se no, esegue il test successivo
MEM.SL      TAY              Memorizza l'accumulatore nell'in-
                             dirizzo selezionato usando Y per
                             contenere il carattere
            LDA TV.PTR       Salva TV.PTR
            PHA              nello stack e in X prima di usarlo
            LDX TV.PTR+1     per visualizzare il carattere nella
                             posizione desiderata
            LDA SELEZ        Setta TV.PTR=SELEZ
            STA TV.PTR       in modo che punti
            LDA SELEZ+1      all'indirizzo selezionato
            STA TV.PTR+1
            TYA              Rimette nell'accumulatore il carat-
                             tere da visualizzare
            LDY #0           Lo visualizza nella posizione
            STA (TV.PTR),Y   desiderata
```

```
         STX TV.PTR+1     Resetta TV.PTR al valore originale
         PLA
         STA TV.PTR
         RTS

IF.GO    CMP #'G          È il tasto G?
         BNE IF.HEX       Se no, esegue il test successivo
GO       LDY REG.Y        Se sì, carica i registri del 6502
         LDX REG.X        con la loro immagine visualizzata
         LDA REG.P
         PHA
         LDA REG.A
         PLP
         JSR CALLSL       Chiama la subroutine all'indirizzo
                          selezionato
         PHP              Al ritorno dalla subroutine
         STA REG.A        salva il valore dei registri
         STX REG.X        nelle rispettive immagini
         STY REG.Y
         PLA
         STA REG.P
         RTS              Ritorna al programma principale
CALLSL   JMP (SELEZ)      Chiama la subroutine all'indirizzo
                          selezionato
IF.HEX   PHA              Salva il carattere
         JSR BINA         Se l'accumulatore contiene un ca-
                          rattere ASCII compreso tra 0 e 9 o
                          A e F, BINA calcola la rappresenta-
                          zione binaria di quel carattere. Al-
                          trimenti BINA setta l'accumulatore
                          a FF e l'indicatore negativo
         BMI IF.CLR       Se l'accumulatore non contiene un
                          carattere esadecimale esegue il test
                          successivo
         TAY
         PLA
         TYA
METTI    LDX CAMPO        Mette A in un campo esadecimale
         BNE NOINDI       L'indice è sotto il campo indirizzi?
                          Se no, l'indice deve essere sotto un
                          altro campo esadecimale
CAMPO1   LDX #3           Poiché l'indice è sotto il campo
LOOP.1   CLC              indirizzi, mette il carattere esadeci-
         ASL SELEZ        male contenuto in A nel campo in-
```

```
        ROL SELEZ+1     dirizzi settando il puntatore che
        DEX             specifica l'indirizzo visualizzato
        BPL LOOP.1
        TYA
        ORA SELEZ
        STA SELEZ
        RTS             Ritorna al programma principale
NOINDI  CPX #1          L'indice è sotto il campo 1?
        BNE CAMPOR      Se no, deve essere sotto l'immagine
                        di un registro
ROLLSL  AND #$0F
        PHA
        JSR LEGBYT      Legge il contenuto dell'indirizzo
                        selezionato e trasla a sinistra
        ASL A           4 volte
        ASL A
        ASL A
        ASL A
        AND #$F0
        STA TEMP        Lo salva in una variabile temporanea
        PLA
        ORA TEMP
        JSR MEM.SL      Memorizza il risultato nell'indirizzo
        RTS             selezionato e ritorna al programma
                        principale
TEMP    .BYTE 0
CAMPOR  DEX             L'indice deve essere sotto il campo
        DEX             3,4, 5 o 6
        DEX
        LDY #3
LOOP.2  CLC             Mette il carattere esadecimale con-
        ASL REGI,X      tenuto in A nell'appropriata imma-
        DEY             gine
        BPL LOOP.2
        ORA REGI,X
        STA REGI,X
        RTS             Ritorna al programma principale
IF.CLR  PLA             Recupera il carattere dallo stack
        CMP #'Q         È una Q? (Q per fine)
        BNE NO.Q        Se no, esegue il test successivo
        PLA             Se sì, ritorna
        PLA             al programma
        PLP             che ha richiamato
        RTS             VISMON
```

NO.Q	JSR COMOD	Cambia questo salto a COMOD con la chiamata ad altre subroutine che estendono la funzionalità
COMOD	RTS	Ritorna al programma principale

Conversione ASCII/binario

La subroutine AGGIOR richiede un'altra subroutine chiamata BINA, che ci permette di stabilire se il carattere contenuto in A è compreso tra 0 e 9 o A e F e, nel caso, di calcolare l'equivalente binario. Se l'accumulatore contiene caratteri ASCII al di fuori dei limiti definiti, BINA fornirà il codice di errore $FF. Perciò:

Accumulatore	BINA
$30 (ASCII "0")	$00
$31 (ASCII "1")	$01
$32 (ASCII "2")	$02
$33 (ASCII "3")	$03
$34 (ASCII "4")	$04
$35 (ASCII "5")	$05
$36 (ASCII "6")	$06
$37 (ASCII "7")	$07
$38 (ASCII "8")	$08
$39 (ASCII "9")	$09
$41 (ASCII "A")	$0A
$42 (ASCII "B")	$0B
$43 (ASCII "C")	$0C
$44 (ASCII "D")	$0D
$45 (ASCII "E")	$0E
$46 (ASCII "F")	$0F
altri valori	$FF

Possiamo risolvere questo problema usando un'apposita tabella che chiameremo BINTAB. BINTAB si trova all'indirizzo $2000, quindi $2000 conterrà $FF come pure $2001, $2002, fino all'indirizzo $202F; questo perché nessuno dei codici ASCII compresi tra $00 e $2F rappresenta i caratteri compresi tra 0 e 9 o A e F. L'indirizzo $2030 conterrà 00, perché $30 (il suo offset nella tabella) rappresenta il codice ASCII dello 0, e in questo modo l'indirizzo $2030 contiene il suo equivalente binario: $00. Similmente, poiché $31 è il carattere ASCII 1, l'indirizzo $2031 conterrà un 1 binario, $01, $2032 conterrà $02 e così via fino a $2039 che conterrà $09.

Gli indirizzi da $203A a $2040 conterranno tutti $FF, perché nessuno dei codici ASCII compresi tra $3A e $40 rappresentano uno dei caratteri compresi tra 0 e 9 o A e F.
Però, l'indirizzo $2041 conterrà $0A perché $41 rappresenta una A in ASCII e $0A è l'equivalente binario. Lo stesso ragionamento vale fino all'indirizzo $2046 che conterrà $0F. Gli indirizzi da $2047 a $20FF conterranno $FF perché nessuno dei valori compresi tra $47 e $FF rappresentano in ASCII i caratteri da 0 a 9 o da A a F.
BINA quindi sarà una routine semplicissima:

```
BINA    TAY             Usa il carattere ASCII come indice
        LDA BINTAB,Y    Trova il valore in tabella
        RTS
```

Questo è un tipico esempio di ricerca in tabella, ma è necessaria una tabella di 256 byte. Forse una routine leggermente più complicata può pervenire agli stessi risultati con una tabella più piccola o addirittura facendone a meno. Una routine di questo tipo dovrà calcolare il risultato piuttosto che cercarlo. Guardiamo più attentamente i caratteri da convertire. Questi caratteri saranno compresi nell'intervallo da $30 a $39 o da $41 a $46. Un input nell'intervallo $30-$39 sarà un carattere ASCII compreso tra 0 e 9 e, sottraendo $30 da questo input, otterremo la conversione. Un input nell'intervallo $41-$46 è un carattere ASCII compreso tra A e F e, sottraendo $37 da questo input otterremo la conversione. Per esempio, $41 (A in ASCII) meno $36 darà $0A (A in binario).
Ogni altro valore esterno a questi intervalli è illegale e verrà convertito in $FF.
Tenendo presente queste relazioni, la subroutine BINA dovrà solo stabilire se il carattere contenuto nell'accumulatore appartiene a uno degli intervalli legali e, se così, eseguire l'appropriata sottrazione.
Qui di seguito viene presentata una subroutine BINA che non richiede l'uso di tabelle:

```
BINA    SEC             Si prepara alla sottrazione
        SBC #$30        Sottrae $30
        BCC NOK         Se il carattere era minore di $30
                        non andava bene, quindi converte
                        in $FF
        CMP #$0A        Era nell'intervallo $30-$39?
        BCC OK          Se sì, andava bene e quindi è già
                        stata effettuata la conversione
        SBC #7          Sottrae 7
        CMP #$10        Era nell'intervallo $41-$46?
        BCS NOK         Se sì, non andava bene
```

```
         SEC
         CMP #$0A
         BCS OK
NOK      LDA #$FF
         RTS
OK       LDX #0
         RTS
```

Le utility del Monitor Visibile

Il Monitor Visibile mette a disposizione le seguenti subroutine:

BINA	Determina se l'accumulatore contiene l'equivalente di un carattere esadecimale. Se sì, esegue la conversione binaria, se no, converte nel valore $FF
CALLSL	Chiama l'indirizzo attualmente selezionato come una subroutine
DEC.SL	Seleziona l'indirizzo precedente decrementando il puntatore SELEZ
LEGKEY	Legge il valore del tasto premuto richiamando la routine della ROM
LEGBYT	Legge il byte contenuto nell'indirizzo selezionato
GO	Carica i registri con i valori visualizzati
INC.SL	Seleziona il byte successivo incrementando il puntatore SELEZ
MEM.SL	Memorizza il contenuto dell'accumulatore nell'indirizzo selezionato
VISMON	Permette all'utente di dare comandi al Monitor Visibile finché l'utente non preme il tasto Q (fine)

La Figura 6.6 mostra la gerarchia delle routine del Monitor Visibile, alcune delle quali saranno presentate più avanti.

Uso del Monitor Visibile

Il Capitolo 13 mostra come è possibile memorizzare il codice oggetto del Monitor Visibile. Per usare questo programma con tutte le caratteristiche qui descritte è necessario usare il programma BASIC di caricamento del Capitolo 13 che ci permetterà di caricare le routine presentate nelle seguenti appendici:

	Appendici	Contenuto
	E1	Utility dello schermo
	E2	Monitor Visibile primo livello
	E3	Monitor Visibile subroutine AGGIOR
VIC-20	E12	Tabella dati di sistema del VIC-20
C-64	E13	Tabella dati di sistema del C-64

Concatenando i programmi oggetto contenuti in queste appendici otterrete un monitor perfettamente funzionante. Ma questo è solo l'inizio. Ci sono molte funzioni che possono essere aggiunte ed è quello che faremo nei prossimi capitoli.

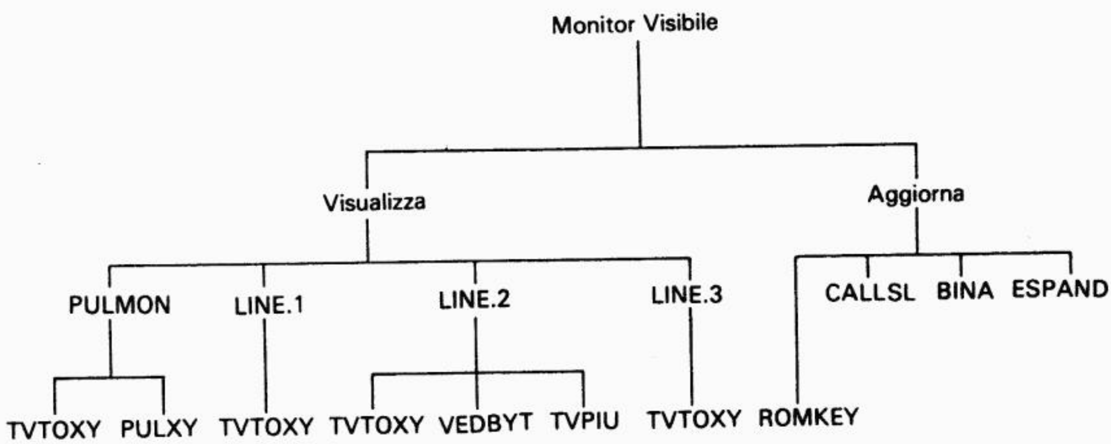


Figura 6.6 La gerarchia delle routine del Monitor Visibile

Capitolo

Utility di stampa 7

Il Monitor Visibile è di grande aiuto per esaminare e modificare il contenuto della memoria, ma, così come è stato sviluppato fino ad ora, è muto, non può cioè "parlarci" se non attraverso la limitata visualizzazione dei campi. Potremo quindi inserire i caratteri in memoria ma non saremo in grado di listarli. Quindi a questo punto dovremo sviluppare delle routine che ci permettano di indirizzare messaggi, testi o stringhe qualsiasi di caratteri ad una qualsiasi periferica di output.
Questo capitolo esaminerà in dettaglio una serie di routine di stampa. Nel sistema operativo del vostro computer sono presenti delle routine che controllano la visualizzazione. Il C-64 e il VIC-20 posseggono alcune routine in ROM che permettono di indirizzare l'uscita al video, simulando quello che viene comunemente chiamato un TVT (*TeleVision Typewriter* - macchina da scrivere video), oppure alla periferica connessa alla porta seriale, generalmente una stampante. Poiché queste routine sono già presenti nel sistema operativo del vostro computer, non ci sarà bisogno di reinventarle, ma piuttosto cercheremo di sviluppare un software che indirizzi i dati da visualizzare all'appropriata routine del sistema operativo.
Siccome ci apprestiamo a scrivere del software che richiama le due routine standard di output, perché non prevedere un aggancio con una routine di output specificatamente scritta dall'utente? Una possibilità del genere renderà estremamente semplice indirizzare l'output (messaggi, dump esadecimali, listati, ecc.) allo schermo, alla stampante o a qualsiasi periferica speciale che potreste avere connesso al vostro sistema.

Selezione della periferica di output

Qualsiasi programma dovrebbe essere in grado di indirizzare i dati allo schermo, e/o alla stampante, e/o a una subroutine-utente. Quello che ci occorre quindi sono alcune subroutine che permettano di selezionare singolarmente o globalmente le periferiche. Chiameremo queste subroutine TVT.ON, TVTOFF, PR.ON, PR.OFF, USR.ON, USROFF, ALL.ON e ALLOFF. Con queste subroutine il programma principale sarà in grado di attivare o disattivare le periferiche singolarmente o globalmente.
La linea che selezionerà il video per l'output sarà:

```
JSR TVT.ON
```

mentre la disattivazione avverrà per mezzo di:

```
JSR TVTOFF
```

Le subroutine di selezione o esclusione faranno uso di tre flag: TVT, PRINTR, USR. Il flag TVT indicherà se lo schermo è selezionato come periferica di uscita; PRINTR avrà la stessa funzione per la stampante e USR per la subroutine definita dall'utente.
Per convenienza useremo byte differenti per ogni flag e definiremo i flag in OFF quando il valore è zero e in ON quando il valore è diverso da zero. Usando questa convenzione potremo selezionare una periferica semplicemente inserendo un valore diverso da zero nel flag di quella periferica e, per escluderla, dovremo inserire uno zero nel flag.
Le definizioni dei flag e il listato delle subroutine di selezione o esclusione sono riportati qui di seguito:

Flag delle periferiche

	OFF=0	Quando un flag ha valore zero la rispettiva periferica non è attiva
	ON=$FF	Quando un flag ha valore $FF la rispettiva periferica è attiva
TVT	.BYTE ON	Questo flag è zero quando il video è escluso. Per default il video è attivo
PRINTR	.BYTE OFF	Questo flag è zero quando la stampante è esclusa. Per default la stampante è esclusa
USR	.BYTE OFF	Questo flag è zero quando la subroutine utente è esclusa. Per default la subroutine è esclusa

Subroutine di selezione/esclusione

TVT.ON	LDA #ON STA TVT RTS	Attiva il video come periferica di ouput settando il flag con il valore $FF
TVTOFF	LDA #OFF STA TVT RTS	Disattiva il video azzerando il flag
PR.ON	LDA #ON STA PRINTR RTS	Attiva la stampante settando il flag con il valore $FF
PR.OFF	LDA #OFF STA PRINTR RTS	Disattiva la stampante azzerando il flag
USR.ON	LDA #ON STA USR RTS	Attiva la routine utente settando il flag a $FF
USROFF	LDA #OFF STA USR RTS	Disattiva la routine utente azzerando il flag
ALL.ON	JSR TVT.ON JSR PR.ON JSR USR.ON RTS	Attiva tutte le periferiche
ALLOFF	JSR TVTOFF JSR PR.OFF JSR USROFF RTS	Disattiva tutte le periferiche

Una routine di stampa dei caratteri

Ora che il programma principale può attivare o disattivare qualsiasi combinazione di periferiche, ci occorre una routine che invii i caratteri alla periferica selezionata. Chiameremo questa routine INVCAR, vista la sua funzione di inviare caratteri alle periferiche.
Tutto il software presentato farà uso di questa routine per inviare i caratteri e in nessun caso userà le routine del sistema operativo direttamente.
Questo fatto renderà il software più facile da aggiornare. Se, per esempio svilupperete differenti routine di output, il programma principale continuerà ad usare INVCAR evitandovi di dover adattare la routine.
Quando INVCAR viene richiamata, verrà eseguito un test sul flag TVT e

se questo flag è settato verrà richiamata la routine di stampa sul video. Lo stesso test verrà eseguito sia per il flag PRINTR che per il flag USR e se anche questi sono settati, i caratteri verranno inviati anche alle rispettive routine. La Figura 7.1 mostra il diagramma di flusso della routine INVCAR.

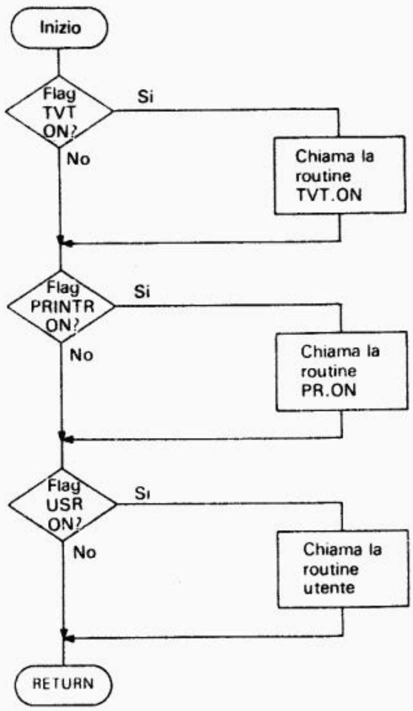


Figura 7.1 Diagramma di flusso della routine INVCAR, per inviare dati a tutte le periferiche selezionate

Vettori di output

Se le routine di output sono situate in diverse posizioni della memoria come sarà possibile far sapere a INVCAR gli indirizzi delle routine che dovrà chiamare? Non sarà possibile direttamente, ma solo indirettamente per mezzo di alcuni puntatori che avremo opportunamente predisposto.
Occorrono per questo scopo tre puntatori, o vettori di output, che chiamino le routine di output del sistema operativo. Il primo puntatore che chiameremo ROMTVT indirizzerà alla routine di output per il video; il secondo, ROMPRT, indirizzerà alla routine di output della porta seriale; il terzo, USROUT, dovrà indirizzare alla routine di output scritta

dall'utente per scopi particolari (se non avete necessità di quest'ultima routine, USROUT indirizzerà una routine fittizia consistente in una singola istruzione: RTS).
Se dovesse capitare per un motivo qualsiasi di dover rilocare in memoria queste routine, sarà sufficiente modificare solo il vettore riguardante la routine interessata.
I puntatori ROMTVT, ROMPRT e USROUT non dovranno necessariamente risiedere vicino a INVCAR, il che significa che potremo memorizzare tutti i vettori e i puntatori riguardanti il nostro sistema in una determinata area di memoria che chiameremo per comodità *dati di sistema*. Per ulteriori informazioni riguardo a questo blocco di memoria basterà fare riferimento alle Appendici B1 o B2.
Riportiamo qui di seguito il listato della routine INVCAR:

INVCAR

```
INVCAR    STA CAR        Salva il carattere
          BEQ FINE       Se non ce n'è ritorna senza stampare
          LDA TVT        È attivo il video?
          BEQ IF.PR      Se no, controlla la periferica suc-
                         cessiva
          LDA CAR        Se sì, indirizza indirettamente il
          JSR VAI. 1     carattere alla routine video
IF.PR     LDA PRINTR     È attiva la stampante?
          BEQ IF.USR     Se no, controlla la periferica suc-
                         cessiva
          LDA CAR        Se sì, indirizza indirettamente il
          JSR VAI. 2     carattere alla routine stampante
IF.USR    LDA USR        È attiva la routine utente?
          BEQ FINE       Se no, salta alla label FINE
          LDA CAR        Se sì, indirizza indirettamente il
          JSR VAI. 3     carattere alla routine utente
FINE      RTS            Ritorna al programma principale
CAR       .BYTE 0        Questo byte contiene l'ultimo ca-
                         rattere inviato
```

Chiamate alle routine

```
VAI. 1    JMP (ROMTVT)
VAI. 2    JMP (ROMPRT)
VAI. 3    JMP (USROUT)
```

Routine di output per caratteri speciali

INVCAR è una routine di output generale; cerchiamo adesso di sviluppare una routine di output che ci permetta di svolgere le funzioni che sono richieste più spesso. Per esempio, si presenta frequentemente la necessità di inviare un a capo (LF-*Line Feed*) e un ritorno carello (CR-*Carriage Return*) per posizionarsi sulla riga successiva, oppure di lasciare spazi bianchi o di dover visualizzare byte in forma esadecimale.
Quello che ci prefiggiamo è sviluppare alcune routine dedicate proprio a queste funzioni. Poiché ognuna di queste subroutine chiamerà INVCAR, i caratteri potranno essere inviati a tutte le periferiche selezionate.
Di seguito sono riportati i listati di queste tre routine:

CR-LF

```
          CR=$0D        Codice ASCII di CR
          LF=$0A        Codice ASCII di LF

CR.LF     LDA #CR       Invia un CR e LF a tutte le perife-
          JSR INVCAR    che selezionate
          LDA #LF
          JSR INVCAR
          RTS
```

SPAZIO

```
SPAZIO    LDA #$20      Carica nell'accumulatore il codice
          JSR INVCAR    ASCII corrispondente allo spazio e
                        lo invia alle periferiche attive
          RTS
```

BYTE

```
PR.BYT    PHA           Salva il byte
          LSR A         Determina l'ASCII corrispondente
          LSR A         ai 4 bit più significativi
          LSR A
          LSR A
          JSR ASCII
          JSR INVCAR    Invia il carattere alle periferiche
                        attive
```

	PLA	Determina l'ASCII dei 4 bit meno significativi
	JSR ASCII	
	JSR INVCAR	Invia il carattere alle periferiche attive
	RTS	

Invio ripetitivo

Poiché alcuni programmi potrebbero avere la necessità di inviare più di uno spazio o CR/LF o altro carattere, perché non sviluppare alcune utility che permettano di inviare lo stesso carattere ripetitivamente alle periferiche?
In qualunque caso sarà necessario memorizzare nel registro X il numero di ripetizioni richieste e poi chiamare la routine SPAZI per inviare X spazi, oppure la routine CRLFX per inviare X a capo, oppure ancora la routine CARX per inviare X volte il carattere contenuto nell'accumulatore. Se il valore del registro X è zero, non verrà inviato alcun carattere. Per inviare sette spazi quindi sarà necessario includere le seguenti linee di programma:

```
LDX #7
JSR SPAZI
```

Per inviare quattro a capo:

```
LDX #4
JSR CRLFX
```

Per inviare dieci asterischi (*):

```
LDX #10
LDA #'*
JSR CARX
```

Per poter utilizzare queste istruzioni ci occorrono tre piccole subroutine: SPAZI, CRLFX e CARX.

SPAZI-CARX

SPAZI	LDA #$20	Carica l'accumulatore con uno spazio
CARX	STX RIPETE	Inizializza il contatore di ripetizione

```
RPLOOP   PHA            Salva il carattere da ripetere
         LDX RIPETE     È stato inviato l'ultimo carattere?
         BEQ FINERI     Se sì, esce dal loop,
         DEC RIPETE     altrimenti decrementa il contatore
         JSR INVCAR     Invia il carattere alle periferiche
                        attive
         PLA
         CLC
         BCC RPLOOP     Ripete il carattere se necessario
FINERI   PLA            Pulisce lo stack
         RTS
```

CRLFX

```
CRLFX    STX RIPETE     Inizializza il contatore
CRLOOP   LDX RIPETE
         BEQ FINECR     Se ha finito ritorna
         DEC RIPETE     Decrementa il contatore
         JSR CR.LF      Invia un a capo
         CLC            Controlla se era l'ultimo
         BCC CRLOOP
FINECR   RTS            Se ha finito ritorna
RIPETE   .BYTE          Questo byte è usato come contatore
```

Come stampare un messaggio

Si potrà presentare la necessità di visualizzare (o stampare) un messaggio contenuto in una arbitraria locazione di memoria. A questo scopo svilupperemo una routine, che chiameremo PR.MSG.
Per mezzo di questa routine potremo inviare un messaggio a tutte le unità periferiche attive. Per poter funzionare sarà necessario prelevare tutti i caratteri che compongono il messaggio ad uno ad uno in maniera sequenziale e passarli alla routine. Anche in questo caso dovremo far sapere alla routine in che locazioni di memoria è presente il messaggio e, come abbiamo operato in precedenza, potremmo fare uso di un puntatore che sia inizializzato volta per volta con la locazione di partenza del messaggio da inviare.
Questa soluzione però richiede due ulteriori byte per il puntatore. Potremo usare un registro per specificare la locazione del puntatore che punta all'inizio del messaggio. Presumibilmente il programma principale è in grado di trovare 2 byte in un'opportuna locazione della pagina zero, an-

che se dovrà salvare il valore originale dei byte scelti prima di settarli. Il programma principale potrà quindi settare questo puntatore in modo che indirizzi all'inizio del messaggio e poi fare in modo che il registro X specifichi questo puntatore di pagina zero. Usando l'indirizzamento indicizzato indiretto, la routine PR.MSG sarà in grado di leggere i caratteri che compongono il messaggio.
Quando tutti i caratteri saranno stati stampati, si tornerà al programma principale.
Come potrà PR.MSG sapere quando ha raggiunto la fine del messaggio? Potremmo segnare la fine del messaggio con un carattere speciale: ETX (*End of TeXt* - fine del testo) e, per ragioni che approfondiremo nel Capitolo 10, segneremo anche l'inizio di ogni messaggio con un carattere speciale: TEX (*TEXt follows* - il testo segue).
Se saremo in grado di sviluppare la routine PR.MSG in modo che lavori con questi tre input, non sarà difficile inviare alle periferiche qualsiasi particolare messaggio contenuto in memoria. Vediamo come dovrà essere la sequenza delle istruzioni; supponiamo di avere, in una certa area di memoria, un messaggio che inizi con TEX e finisca con ETX e di chiamare il byte alto dell'indirizzo MSG.AL e il byte basso MSG.BA. Quindi, se il messaggio inizia all'indirizzo $13A9 il byte alto sarà MSG.AL=$13 e quello basso MSG.BA=$A9, MSGPTR è un puntatore di pagina zero posizionabile ovunque in questa area di memoria. Se il programma principale non ha necessità di conservare MSGPTR, sarà possibile visualizzare il messaggio sullo schermo usando le seguenti istruzioni:

```
JSR TVT.ON         Attiva il video
LDA #MSG.BA        Setta MSGPTR in modo che
STA MSGPTR         punti all'inizio del messaggio
LDA #MSG.AL
STA MSGPTR+1
LDX #MSGPTR        Setta il registro X in modo che punti a
                   MSGPTR
JSR PR.MSG         Invia il messaggio alle periferiche attive
```

Se invece il programma principale deve conservare MSGPTR, sarà necessario salvare MSGPTR e MSGPTR+1 prima di eseguire la routine di cui sopra e resettarli dopo l'esecuzione. Questa sembra essere la maniera più conveniente per ottenere il nostro scopo. Adesso volgiamo l'attenzione alla routine PR.MSG e cerchiamo di svilupparla in modo che soddisfi le caratteristiche richieste.

PR.MSG

PR.MSG	STX TEMP.X	Salva il registro X che specifica il puntatore
	LDA 1,X	Salva il puntatore
	PHA	
	LDA 0,X	
	PHA	
LOOP	LDX TEMP.X	Resetta X con il valore originale in modo che indichi il puntatore
	LDA (0,X)	Legge il carattere successivo del messaggio
	CMP #ETX	È la fine del messaggio?
	BEQ FINMSG	Se sì, salta a FINMSG
	INC 0,X	Se no, incrementa il puntatore
	BNE NEXT	in modo che indichi il carattere
	INC 1,X	successivo del messaggio
NEXT	JSR INVCAR	Invia il carattere alle periferiche attive
	CLC	Legge il carattere successivo
	BCC LOOP	del messaggio
FINMSG	PLA	Resetta il puntatore
	STA 0,X	
	PLA	
	STA 1,X	
	RTS	Ritorna con MSGPTR salvato
TEMP.X	.BYTE 0	Questo spazio è usato per conservare il valore iniziale di X

Come stampare un testo

Sarebbe molto più conveniente una routine che non richiede l'uso di puntatori o registri per indicare la locazione di un messaggio; ma, se non esiste un puntatore o un registro che indichi la locazione di partenza di un messaggio, come potrà saperlo la subroutine? Guardando nello stack.
Cerchiamo di studiare una subroutine, che chiameremo "invia il testo seguente alla periferica selezionata", che invii il messaggio. Poiché il nome è più lungo di sei caratteri, chiameremo questa routine INVIA: (il carattere ":" vuole suggerire "il testo seguente"). Il programma principale potrebbe quindi inviare il messaggio "CIAO" usando la routine presentata qui di seguito:

```
JSR TVT.ON        Attiva il video
JSR INVIA:
.BYTE TEX
.BYTE "CIAO"
.BYTE ETX
```

Quando il 6502 richiama una subroutine, memorizza l'indirizzo di ritorno nello stack. Questo fatto permette, quando la subroutine finisce (RTS), di ritornare esattamente al punto di partenza, perché il 6502 sa di poter trovare l'indirizzo di ritorno nello stack.
La subroutine INVIA: può sfruttare questa caratteristica togliendo il suo indirizzo di ritorno dallo stack e usandolo come puntatore per il messaggio da stampare. Raggiunta la fine del messaggio, metterà il nuovo indirizzo di ritorno nello stack, un indirizzo che punta alla fine del messaggio. Quindi INVIA: può eseguire un RTS e il controllo verrà passato all'istruzione immediatamente successiva all'ETX che segna la fine del messaggio. Di seguito riportiamo il listato sorgente alla routine INVIA:

```
INVIA:    PLA              Estrae l'indirizzo di ritorno
          TAX              dallo stack e lo salva nei
          PLA              registri X e Y
          TAY
          JSR SAL.SL       Salva il puntatore di ritorno perché
                           verrà usato come puntatore di mes-
                           saggio
          STX SELEZ        SELEZ diventa l'indirizzo di ritorno
          STY SELEZ+1
          JSR INC.SL       Incrementa SELEZ in modo che
                           punti al carattere successivo
NEXTCA    JSR INC.SL
          JSR LEGBYT       Legge il carattere
          CMP #ETX         È l'indicatore di fine testo?
          BEQ STOP         Se sì, calcola l'indirizzo di ritorno e
                           ritorna
          JSR INVCAR       Se no, invia il carattere alle perife-
                           riche attive
          CLC              Legge il carattere successivo
          BCC NEXTCA
STOP      LDX SELEZ
          LDY SELEZ+1
          JSR REC.SL       Resetta il puntatore al valore
                           iniziale
          TYA              Mette l'indirizzo dell'ETX nello
          TXA              stack
```

```
          PHA
          RTS               Ritorna due byte immediatamente
                            dopo l'ETX
```

Salvataggio e recupero del puntatore SELEZ

Ora che un certo numero di routine accede alla memoria per mezzo delle utility di SELEZ (LEGBYT, MEM.SL, INC.SL e SAL.SL) dobbiamo ancora sviluppare un paio di subroutine che permettano di salvare e recuperare il puntatore SELEZ. Con queste funzioni di salvataggio e recupero ogni subroutine potrà fare uso del puntatore per accedere alla memoria senza interferire con le altre subroutine che usano lo stesso puntatore.
SAL.SL e REC.SL rispettivamente salvano e recuperano SELEZ per mezzo dello stack e, inoltre, non alterano i registri X e Y e la pagina zero.

Salva SELEZ

```
SAL.SL    PLA               Legge l'indirizzo di ritorno dallo
                            stack
          STA RETURN        e lo mette in RETURN
          PLA
          STA RETURN+1
          LDA SELEZ+1       Mette SELEZ nello stack
          PHA
          LDA SELEZ
          PHA
          LDA RETURN+1      Rimette nello stack l'indirizzo di
                            ritorno
          PHA
          LDA RETURN
          PHA
          RTS               Ritorna al programma principale
```

Recupera SELEZ

```
REC.SL    PLA               Salva temporaneamente l'indirizzo
                            di ritorno
          STA RETURN
          PLA
          STA RETURN+1
```

```
           PLA                Recupera SELEZ dallo stack
           STA SELEZ
           PLA
           STA SELEZ+1
           LDA RETURN+1       Rimette l'indirizzo di ritorno nello
                              stack
           PHA
           LDA RETURN
           PHA
           RTS
RETURN     .BYTE 0            Puntatore usato da SAL.SL e
                              REC.SL per preservare il loro in-
                              dirizzo di ritorno
```

Conclusioni

Con le utility di stampa presentate in questo capitolo, dovrebbe risultare facile scrivere la porzione di stampa per ogni programma, con possibilità di selezionare ogni possibile combinazione di periferiche di output e di inviare a queste un carattere, un messaggio completo o un byte.
Il programma principale sarà completamente indipendente dalla particolare rappresentazione dei dati usata dalle utility. Notate inoltre che le utility di stampa non operano mai direttamente sul puntatore SELEZ, essendo completamente isolate dalla natura e dalla locazione di questo puntatore.
Le utility di stampa, conoscendo la locazione delle subroutine di SELEZ, saranno in grado di leggere il byte selezionato, selezionare quello precedente o successivo, salvare il puntatore nello stack e recuperarlo.
Le utility di stampa presentate in questo capitolo sono le seguenti:

CARX	Invia il carattere contenuto nell'accumulatore X volte alla periferica selezionata
CR.LF	Invia un a capo alla periferica selezionata
CR.LFX	Invia X a capo alla periferica selezionata
PR.BYT	Invia il byte contenuto nell'accumulatore in rappresentazione esadecimale
PR.CAR	Invia il carattere contenuto nell'accumulatore alle periferiche attive
PR.MSG	Invia il messaggio indicato da un puntatore in pagina zero definito dal registro X
INVIA:	Invia il messaggio memorizzato in locazioni successive alla riga che ha richiamato la subroutine

SPAZIO	Invia uno spazio alle periferiche attive
SPAZI	Invia spazi alle periferiche attive

Alcuni esercizi

1. Scrivere un programma per provare la stampante e che invii tutti i caratteri possibili da $00 a $FF.
2. Riscrivere il programma di prova in modo che stampi un carattere per linea.

Capitolo

Il dump esadecimale 8

Il Monitor Visibile ci permette di esaminare la memoria, ma solo un byte alla volta. Vi renderete presto conto che occorre poter disporre di uno strumento software che permetta di visualizzare o stampare un intero blocco di memoria: questo strumento è generalmente chiamato *dump*.
Un dump è di grande utilità quando si devono ricercare eventuali errori nell'ambito dei programmi. Non è possibile correggere i programmi se non si è certi dell'effettivo contenuto della memoria. Un dump sarà in grado di mostrare esattamente cosa è stato inserito nel computer, visualizzando il contenuto della memoria in forma esadecimale.
In questo capitolo svilupperemo due programmi di dump, ognuno per un differente tipo di periferica. Quando si lavora alla tastiera infatti, sarà più comodo poter visualizzare il contenuto della memoria sullo schermo, una linea o un blocco per volta, mentre invece per conservare una documentazione o per sviluppare nuovo software sarà utile poter disporre del dump su carta.
La maggior parte delle istruzioni richieste dalle due routine di stampa sarà identica, sia che venga utilizzato il video, sia che venga utilizzata la stampante. Esistono però sufficienti differenze per giustificare le due routine per il video e per la stampante, che chiameremo rispettivamente TVDUMP e PRDUMP.

TVDUMP

TVDUMP dovrà essere molto flessibile: quando si usa il Monitor Visibile è molto comodo poter visualizzare una o più linee. Come potrà TVDUMP sapere quali linee visualizzare? Poiché il Monitor Visibile permette di selezionare un indirizzo di memoria inserendo caratteri esadecimali nel campo indirizzi, oppure permette di avanzare o retrocedere in memoria, potremo fare in modo che TVDUMP visualizzi il contenuto della memoria partendo dall'indirizzo attualmente selezionato.
Poiché stiamo basando TVDUMP sull'indirizzo selezionato dal Monitor Visibile, potremo usare alcune subroutine del monitor per operare sugli indirizzi. LEGBYT leggerà il byte selezionato e INC.SL incrementerà il puntatore SELEZ in modo da poter selezionare il byte successivo. Le utility TVT.ON e PR.BYT ci permetteranno rispettivamente di attivare lo schermo come periferica di uscita e visualizzare il contenuto dell'accumulatore in forma esadecimale. Vorremmo inoltre poter fare in modo che TVDUMP fornisca delle videate facilmente leggibili perfino su un display di 22 o 40 colonne. Opereremo quindi in modo da poter ottenere due formati di visualizzazione: il primo spezzerà la linea esadecimale (16 byte) in due linee da otto byte e il secondo la dividerà in quattro linee da quattro byte ciascuna. La Figura 8.1 mostra i due formati:

Formato A:

```
0200   HH  HH  HH  HH  HH  HH  HH  HH  HH
0208   HH  HH  HH  HH  HH  HH  HH  HH  HH

0210   HH  HH  HH  HH  HH  HH  HH  HH  HH
0218   HH  HH  HH  HH  HH  HH  HH  HH  HH

-----------------32 colonne-----------------
```

Formato B:

```
0200   HH  HH  HH  HH
0204   HH  HH  HH  HH
0208   HH  HH  HH  HH
020C   HH  HH  HH  HH

0210   HH  HH  HH  HH
0214   HH  HH  HH  HH
0218   HH  HH  HH  HH
021C   HH  HH  HH  HH

------17 colonne------
```

Figura 8.1 I due formati di TVDUMP

Un modo di ottenere questi due formati è visibile nel diagramma di flusso in Figura 8.2. Usando questo diagramma come guida, svilupperemo le routine per ottenere le funzioni richieste.

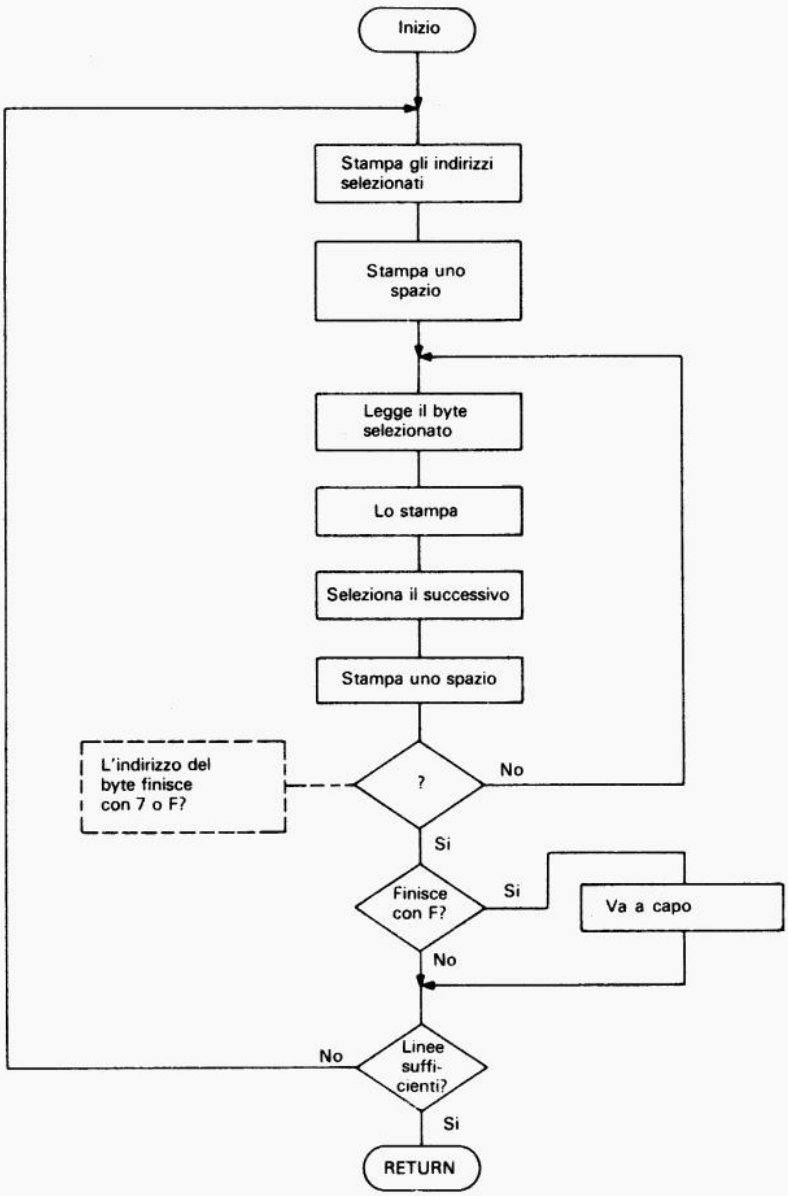


Figura 8.2 Diagramma di flusso della routine di dump

Costanti

CR=$0D	Ritorno carrello
LF=$0A	A capo

Subroutine necessarie

LEGBYT	Legge il byte selezionato
INC.SL	Incrementa il puntatore che specifica l'indirizzo
PR.BYT	Invia il contenuto dell'accumulatore, in forma esadecimale, alle periferiche selezionate
SELEZ	Puntatore dell'indirizzo selezionato

Variabili

CONTAT	.BYTE 0	Questo byte conta il numero di linee visualizzate
MASC	.BYTE 7	Per il formato A (C-64). Per il formato B (VIC-20) usare .BYTE 3

TVDUMP

TVDUMP	JSR TVT.ON	Seleziona il video
	LDA #4	Setta il contatore con il numero di linee
	STA CONTAT	da visualizzare
	LDA SELEZ	Posiziona SELEZ all'inizio di una linea esadecimale (16 byte)
	AND #$FO	
	STA SELEZ	
DUMPLN	JSR PR.ADR	Visualizza l'indirizzo
	JSR SPAZIO	
DMPBYT	JSR SPAZIO	Fa due spazi
	JSR DUMPSL	Visualizza il byte selezionato
	JSR INC.SL	Seleziona il byte successivo
	LDA SELEZ	L'inizio riga è sul video?
	AND MASC	
	BNE DMPBYT	Se no, visualizza il successivo byte
	JSR CR.LF	Se sì, sposta sulla linea successiva
	LDA SELEZ	È l'inizio di una nuova linea esadecimale?
	AND #$0F	
	BNE IFFINE	
	JSR CR.LF	Se sì, salta una linea sullo schermo
IFFINE	DEC CONTAT	Era l'ultima linea esadecimale?
	BNE DUMPLN	Se no, visualizza la successiva
	JSR TVTOFF	Se sì, disattiva il video e
	RTS	ritorna al programma principale

DUMPSL

Questa subroutine legge il byte selezionato da SELEZ e lo visualizza in forma esadecimale su tutte le periferiche attive.

```
DUMPSL   JSR LEGBYT     Legge il byte selezionato
         JSR PR.BYT     Lo visualizza in forma esadecimale
         RTS            Ritorna al programma principale
```

PR.ADR

Questa subroutine visualizza, su tutte le periferiche attive, l'indirizzo selezionato (cioè il valore di SELEZ).

```
PR.ADR   LDA SELEZ+1    Legge il byte alto di SELEZ
         JSR PR.BYT     Lo visualizza
         LDA SELEZ      Legge il byte basso
         JSR PR.BYT     Lo visualizza
         RTS            Ritorna al programma principale
```

PRDUMP

Con la subroutine appena presentata sarà possibile visualizzare il contenuto della memoria, ma come potremo ottenerne la stampa? Il programma di stampa sarà diverso da quello di visualizzazione? Potremo selezionare la stampante e lasciare il video inattivo mantenendo tutto il resto inalterato? Le risposte a queste domande sono tutte affermative, ma così non sarebbe possibile sfruttare tutte le caratteristiche della stampante. TVDUMP produce una rappresentazione facilmente leggibile su un video, che è limitato a 22 o 40 colonne, ma molte stampanti possono stampare 80 o più colonne, e noi cercheremo di trarre vantaggio da questo fatto. Inoltre dobbiamo considerare le differenze operative tra un video e una stampante; se si opera su video non è di nessuna importanza dover premere un tasto per visualizzare una linea, ma usando una stampante sarebbe più comodo poter specificare all'inizio l'area da stampare e poi non preoccuparsene più.
Quando la routine verrà richiamata, pulirà lo schermo e visualizzerà il messaggio "DUMP ESADECIMALE". Poi dovrà richiedere gli indirizzi di partenza e di fine della memoria da stampare.
Una volta acquisiti questi dati, PRDUMP dovrà stampare un dump esadecimale dell'area di memoria richiesta e, per comodità, un titolo per ogni colonna e l'indirizzo di ogni linea (vedi Appendice D).

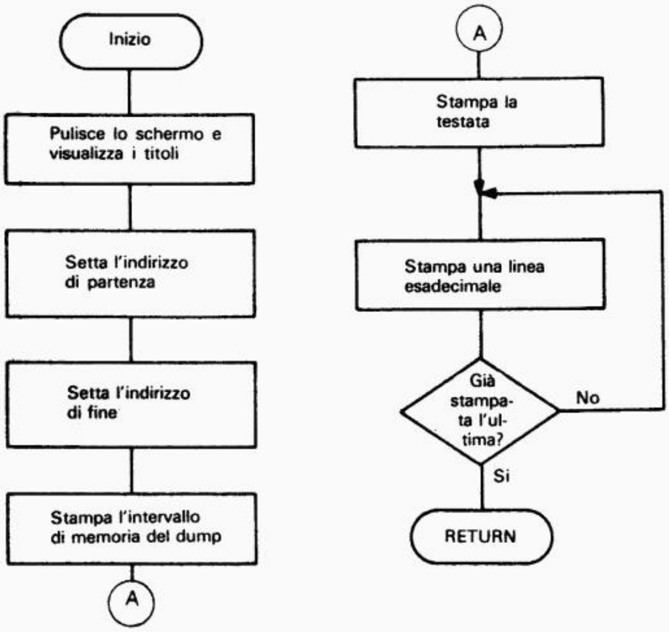


Figura 8.3 Stampa dump

Usando il diagramma di flusso di Figura 8.3 possiamo scrivere il codice sorgente di PRDUMP:

```
PRDUMP   JSR TITOL                     Visualizza il titolo
         JSR SETADS                    Setta gli indirizzi da
                                       stampare
         JSR GOTOSA                    Setta SELEZ=SA (start)
         JSR PR.ON                     Attiva la stampante
         JSR TSTATA                    Stampa la testata
HXLOOP   JSR PRLINE                    Stampa una linea
         BPL HXLOOP                    Finito? Se no, stampa
                                       la successiva
         JSR CR.LF                     Se sì ... nuova riga
         JSR PR.OFF                    Disattiva la stampante
         RTS
TITOL    JSR PUL.TV                    Pulisce lo schermo
         JSR TVT.ON                    Seleziona il video
         JSR INVIA:                    Visualizza il messaggio
         .BYTE TEX                     Inizio testo
         .BYTE CR, 'DUMP '
         .BYTE 'ESADECIMALE',CR
         .BYTE LF,LF,
         .BYTE ETX                     Fine testo
         RTS
```

Indirizzi di partenza e fine

È necessario che la routine abbia ricevuto gli indirizzi di partenza e fine del blocco di memoria da stampare. La subroutine SETADS compirà queste funzioni: visualizzerà sullo schermo i messaggi appropriati e permetterà all'utente di specificare questi indirizzi.
Visualizzare una richiesta sullo schermo è facile: basta attivare il video chiamando TVT.ON, poi chiamare INVIA: seguita dal testo delle richieste. Come potremo specificare gli indirizzi? Sarà necessaria una routine, LEGADR, che legga un indirizzo permettendo all'utente di settare alcuni puntatori.
Questo è fattibile sfruttando il Monitor Visibile, infatti SETADS chiamerà il monitor e al ritorno il puntatore SELEZ indicherà l'indirizzo richiesto.

SETADS

```
SETADS   JSR TVT.ON        Seleziona il video
         JSR INVIA:        Visualizza le richieste
         .BYTE TEX
         .BYTE CR,LF,LF
         .BYTE             'INDIRIZZO DI PARTENZA? (poi
                           premere Q)'
         .BYTE ETX
         JSR VISMON        Chiama il Monitor Visibile per
                           specificare l'indirizzo
         JSR SAQUI         Setta l'indirizzo di partenza
SET.EA   ISR INVIA:        Visualizza la richiesta
         .BYTE TEX
         .BYTE CR,LF,LF
         .BYTE             'INDIRIZZO FINALE? (poi preme-
         .BYTE             re Q)'
         .BYTE ETX
         JSR VISMON        Chiama il monitor per specificare
                           l'indirizzo
         SEC               Testa la validità dei dati
         LDA SELEZ+1
         CMP SA+1
         BCC TROBAS
         BNA EAQUI
         LDA SELEZ
         CMP SA
         BCC TROBAS
```

```
EAQUI     LDA SELEZ+1       EA=SELEZ
          STA EA+1
          LDA SELEZ
          STA EA
          RTS
SAQUI     LDA SELEZ+1       SA=SELEZ
          STA SA+1
          LDA SELEZ
          STA SA
          RTS
TROBAS    JSR INVIA:        Visualizza il messaggio di errore
          .BYTE TEX
          .BYTE CR,LF,LF
          .BYTE             'ERRORE. L'INDIRIZZO FINALE'
          .BYTE             'È MINORE DI QUELLO DI PAR-
          .BYTE             TENZA'
          .BYTE             'CHE È:'
          .BYTE ETX
          JSR PR.SA         Visualizza l'indirizzo
          JMP SET.EA        Richiede l'indirizzo finale
SA        .WORD 0           Puntatore dell'indirizzo di
                            partenza
EA        .WORD $FFFF       Puntatore dell'indirizzo finale
```

A questo punto, in cui l'utente può specificare gli indirizzi di partenza e di fine, ci occorre una routine che stampi l'indirizzo di partenza, l'indirizzo finale o l'area di memoria specificata. Se l'utente specifica la partenza a $D000 e la fine a $D333, dovremo essere in grado di visualizzare $D000, $D333 e $D000-$D333.
Chiameremo queste routine PR.SA, PR.EA e INTERV.

PR.SA

```
PR.SA     LDA #'$           Visualizza il carattere $
          JSR INVCAR        per indicare 'esadecimale'
          LDA SA+1          Byte alto dell'indirizzo
          JSR PR.BYT        di partenza
          LDA SA            Byte basso
          JSR PR.BYT
          RTS
```

PR.EA

```
PR.EA     LDA #'$          Visualizza il carattere $
          JSR INVCAR       per indicare 'esadecimale'
          LDA EA+1         Byte alto dell'indirizzo
          JSR PR.BYT       finale
          LDA EA           Byte basso
          JSR PR.BYT
          RTS
```

INTERV

```
INTERV    JSR PR.SA        Visualizza la partenza
          LDA #'-          Visualizza un trattino
          JSR INVCAR
          JSR PR.EA        Visualizza la fine
          RTS
```

La testata

Ci occorre ora una routine che stampi una testata. Questa testata dovrà specificare il blocco di memoria che si sta stampando e visualizzare un carattere esadecimale in cima ad ogni colonna. Il risultato dovrebbe essere come segue:

```
DUMP HHHH-HHHH

        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
```

(Vedere i dump delle Appendici D). Di seguito riportiamo le poche istruzioni necessarie per stampare la testata.

```
JSR INVIA:
.BYTE TEX,CR,LF
.BYTE 'DUMP'
.BYTE ETX
JSR INTERV
JSR CR.LF
```

Il resto della testata sarà stampato da una subroutine che conta da $0 a $F, converte questi valori nel corrispondente ASCII e li stampa.

Stampa numeri esadecimali (versione 1)

```
         LDX #7          Stampa 7 spazi
         JSRA SPAZI
         LDA #0          Inizializza il contatore di colonne
         STA COLONN      a zero
HXLOOP   LDA COLONN      Converte il contatore in ASCII e
         JSR ASCII       lo stampa
         JSR INVCAR
         LDX #2          Due spazi dopo ogni carattere
         JSR SPAZI
         INC COLONN      Incrementa il contatore
         LDA COLONN      Prosegue se il contatore è minore
         AND #$F0        di $0F,
         BEQ HXLOOP
         LDX #2          altrimenti lascia due righe bianche
         JSR CR.LFX
         RTS             Ritorna al programma principale
COLONN   .BYTE 0         Variabile composta da 1 byte usata
                         come contatore da $00 a $0F
```

Questa prima versione della routine è lunga solamente 49 byte. Ma cosa succederebbe se per qualche ragione uno di questi byte fosse sbagliato? Probabilmente la routine andrebbe direttamente fuori pista. È possibile scrivere questa routine in modo più comprensibile, anche a costo di occupare un maggior numero di byte. Nella versione due della routine sarà necessario chiamare la routine INVIA: e far seguire la testata che si vuole stampare.

Stampa numeri esadecimali (versione 2)

```
JSR INVIA:
.BYTE TEX
.BYTE '      0   1   2   3   4   5   6   7   '
.BYTE        '8  9   A   B   C   D   E   F'
.BYTE CR,LF,LF
.BYTE ETX
RTS
```

Questa seconda versione impiega 60 byte; è molto più comprensibile della prima versione ed inoltre più facilmente modificabile. Non ci sarà bisogno di calcolare indirizzi di salto o controllare il valore di un contatore per determinare se il lavoro è stato svolto. Questa routine ci fornisce un

esempio dei due modi possibili di affrontare il problema: il primo concentrato sui calcoli, mentre il secondo più orientato verso la manipolazione dei dati.
Dove gli altri fattori sono più o meno uguali, è preferibile usare questo secondo modo, perché i programmi risulteranno molto più facili da interpretare e saranno molto più facilmente modificabili. Per questo motivo nei listati della routine TSTATA apparirà questa seconda versione (vedi Appendice C5).

PRLINE

Chiaramente, la maggior parte del lavoro che PRDUMP dovrà compiere, verrà svolto dalla routine PRLINE, che stamperà una linea di memoria sulla stampante. Si fermerà dopo aver stampato 16 byte (cioè una linea), oppure dopo essere arrivata all'indirizzo di fine specificato dall'utente.
Come per TVDUMP useremo SELEZ come puntatore al primo byte che PRLINE dovrà stampare. La prima operazione che PRLINE dovrà eseguire, sarà quella di controllare se il byte selezionato tramite SELEZ è all'inizio di una linea esadecimale. Un byte è all'inizio di una linea esadecimale quando i 4 LSB sono a zero; perciò $4ED8 non è all'inizio di una linea, mentre $4ED0 lo è.
Se il byte selezionato non si trova all'inizio di una linea, la routine PRLINE, dovrà posizionarsi alla colonna di appartenenza del byte, mentre se il byte è all'inizio di una linea esadecimale, la routine, dovrà stampare l'indirizzo del byte scelto e lasciare due spazi.
Una volta raggiunta l'esatta posizione, PRLINE, dovrà leggere il byte scelto, stamparlo in forma esadecimale, saltare di uno spazio e ripetere tutto il procedimento per il byte successivo fino a che tutta la linea è stata stampata o è stato raggiunto l'indirizzo di arrivo specificato dall'utente.
La Figura 8.4 rappresenta il diagramma di flusso della routine.

PRLINE

```
PRLINE    JSR CR.LF        Salta una riga sulla stampante
          LDA SELEZ        Determina la posizione di partenza
          PHA
          AND #$0F
          STA COLONN       Adesso COLONN contiene il nume-
                           ro della colonna dove iniziare la
                           stampa
          PLA              Setta SELEZ all'inizio della linea
          AND #$F0         esadecimale
```

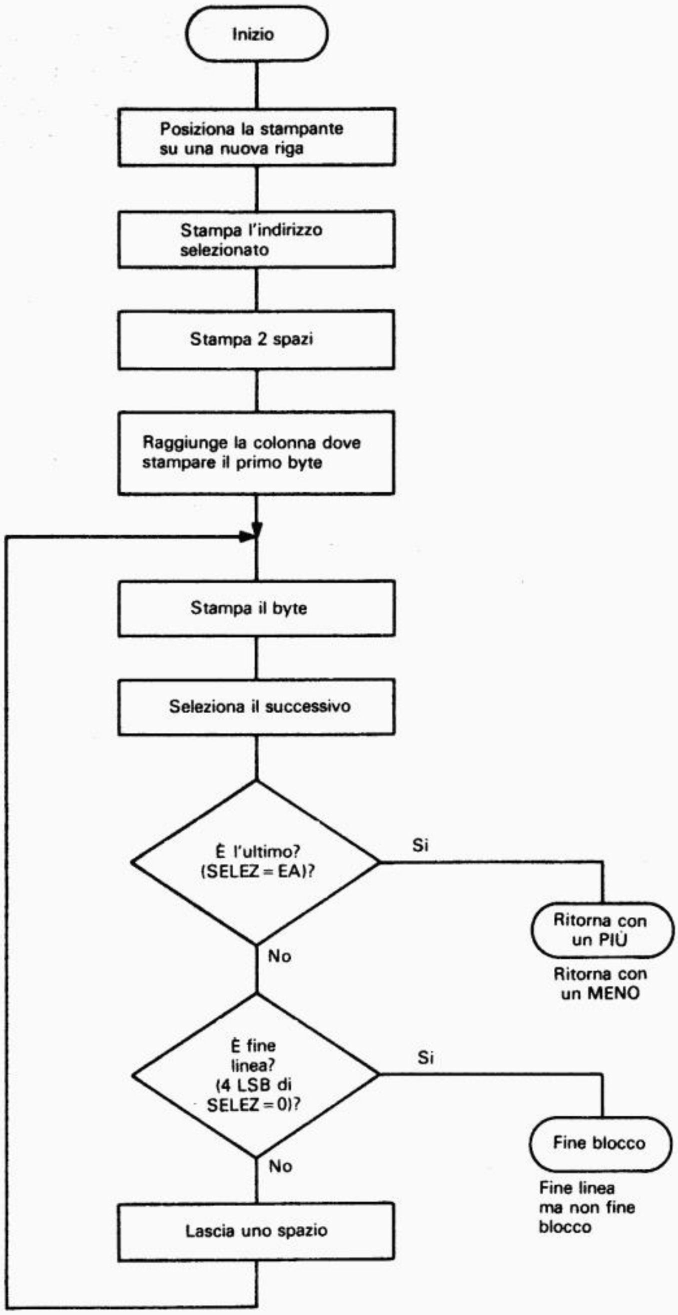


Figura 8.4 Dump di una linea esadecimale sulla stampante

```
            STA SELEZ
            JSR PR.ADR     Stampa l'indirizzo selezionato
            LDX #3         Lascia 3 spazi
            JSR SPAZI
            LDA COLONN     Stampa nella prima colonna?
            BEQ COL.OK     Se sì, la posizione è giusta
LOOP        LDX #3         Se no, incrementa la posizione di 3
            JSR SPAZI      spazi per ogni byte da saltare
            JSR INC.SL
            DEC COLONN
            BNE LOOP
COL.OK      JSR  DUMPSL    Stampa il byte selezionato
            JSR SPAZIO     Lascia uno spazio
            JSR NEXTSL     Seleziona il byte successivo se non
            BMI FINE       siamo arrivati all'indirizzo di fine
NOT.EA      LDA SELEZ      È finita la linea?
            AND #$0F       (4 LSB di SELEZ=0?)
            CMP #0         Se sì, siamo a fine linea
            BNE COL.OK     Se no, seleziona il byte successivo
FINE        RTS
```

Come selezionare il byte successivo (NEXTSL)

NEXTSL controlla se SELEZ è minore dell'indirizzo di arrivo. Se è minore, SELEZ viene incrementato e la subroutine ritorna un PIÙ (con zero nell'accumulatore), altrimenti conserva SELEZ e ritorna con un MENO ($FF nell'accumulatore).

NEXTSL

```
NEXTSL      SEC            Si prepara a confrontare
            LDA SELEZ+1    Il byte alto di SELEZ è minore del
            CMP EA+1       byte alto di EA?
            BCC SL.OK      Se sì, SELEZ può essere incremen-
                           tato
            BNE NO.INC     Se no, non incrementa, perché
                           SELEZ è nella stessa pagina di EA
            SEC            Si prepara a comparare il byte
                           basso
            LDA SELEZ
            CMP EA
```

```
            BCS NO.INC      Se non è minore non incrementa
SL.OK       JSR INC.SL      Poiché SELEZ è minore, possiamo
                            incrementarlo
            LDA #0          Setta il codice di ritorno "PIÙ"
            RTS
NO.INC      LDA #$FF        Setta il codice di ritorno "MENO"
            RTS
```

GOTOSA

GOTOSA setta SELEZ=SA in modo da selezionare il primo byte del blocco definito tra SA ed EA.

```
GOTOSA      LDA SA          Setta SELEZ uguale all'indirizzo
            STA SELEZ       di partenza
            LDA SA+1
            STA SELEZ+1
            RTS
```

A questo punto i due strumenti per il dump sono completi. Sarà possibile richiamarli direttamente dal Monitor Visibile selezionando l'indirizzo di partenza della routine di dump desiderata e digitando G. Questo procedimento andrà bene per PRDUMP; ci saranno le richieste degli indirizzi di partenza e arrivo del blocco da stampare e il dump apparirà sia sul video che sulla stampante. Volendo usare lo stesso procedimento per TVDUMP, quello che otterremo sarà il dump della routine stessa perché questa routine effettua il dump dall'indirizzo visualizzato. Siccome è necessario visualizzare l'indirizzo di partenza per far partire una routine dal Monitor Visibile, non potremo ottenere altro che il dump della routine stessa. Per potere ottenere il dump su video di altre aree di memoria sarà necessario assegnare al Monitor Visibile un tasto (H, per esempio) che richiami la subroutine TVDUMP. Per ulteriori informazioni si rimanda al Capitolo 12 "Il Monitor Visibile integrato".

Capitolo

Un disassemblatore 9

Con il Monitor Visibile possiamo inserire il codice oggetto nella memoria del computer. Con i due strumenti per il dump possiamo visualizzare su video o su carta il contenuto della memoria. Malgrado questo però non possiamo essere sicuri di aver inserito le istruzioni richieste se non comparando i dump con i codici dell'Appendice A3 che contiene la lista degli opcode del 6502. Bisognerebbe verificare che ogni opcode inserito corrisponda all'istruzione e al modo di indirizzamento richiesto. Dovete contare avanti e indietro in esadecimale per controllare se i salti vengono effettuati alle giuste locazioni.

Un disassemblatore (il contrario di assemblatore) può rendere la questione meno spigolosa visualizzando o stampando lo mnemonico corrispondente all'opcode e mostrando l'indirizzo e il modo di indirizzamento rappresentati dall'operando. Il disassemblatore naturalmente non potrà sapere che l'indirizzo $FB ha per label il nome di TV.PTR, ma potrà farvi sapere che una certa istruzione si riferisce all'indirizzo $FB.

Una linea del disassemblatore è composta dai seguenti campi:

Campo	**Descrizione**
1.	Mnemonico
2.	Operando
3.	Indirizzo dell'opcode
4.	Opcode in esadecimale
5.	Primo byte dell'operando (se presente)
6.	Secondo byte dell'operando (se presente)

Una linea disassemblata si presenterà come segue:

1	2	3	4	5	6	(campi)
JSR	0400	08AC	20	00	04	(linea disassemblata)

Come per il dump sarà conveniente prevedere due disassemblatori: uno per il video e l'altro per la stampante. Il primo visualizzerà un certo numero di linee sul video, mentre l'altro chiederà l'indirizzo di partenza e di fine del blocco di memoria da disassemblare.
Sia che si visualizzi su video o si stampi su stampante, il programma disassemblerà una linea alla volta. Ma come è possibile disassemblare una linea di programma? Nello stesso modo in cui lo fareste voi e cioè confrontando l'opcode di un indirizzo di memoria con la tabella dell'Appendice A3. Ogni operazione ha due elementi: uno mnemonico e un modo di indirizzamento. La procedura è semplice: infatti basta scrivere lo mnemonico e poi determinare per mezzo del modo di indirizzamento se questo opcode richiede un operando di un byte, di due byte o se non c'è operando. Se c'è un operando, bisogna guardare i byte successivi e scrivere l'operando dello mnemonico.
Per esempio, se volete disassemblare il programma oggetto memorizzato in una certa locazione, e in quella locazione trovate $8D, consultando la tabella scoprirete che $8D rappresenta STA in modo assoluto. Quindi dovreste cominciare a scrivere STA. Il modo assoluto richiede un operando

Disassemblato

```
JSR   0400      1E00  20  00  04
JSR   04A0      1E03  20  A0  04
LDA   (0021),Y  1E06  B1  21
CLC             1E08  18
BCC   1E00      1E09  90  F5
```

Dump esadecimale

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

1E00  20 00 04 20 A0 04 B1 21 18 90 F5
```

Figura 9.1 Disassemblato e dump esadecimale dello stesso codice macchina

di due byte, così, guardando i due byte successivi potrete trovare ad esempio \$36 e \$D0. La linea disassemblata sarebbe:

STA \$D036

certamente più facile da interpretare che non:

8D 36 D0

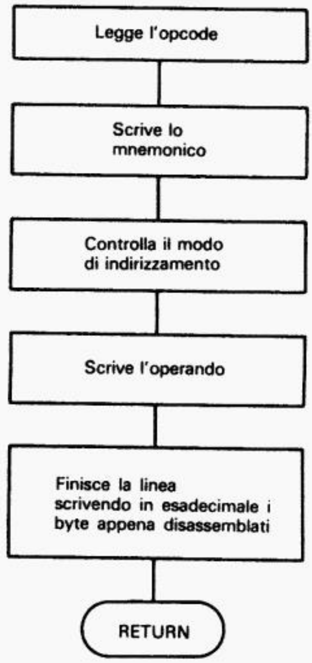


Figura 9.2 Diagramma di flusso per disassemblare una linea di codice macchina

Il procedimento sembra semplice. Possiamo usare SELEZ per indicare il corrente byte in memoria e, assumendo che le routine dei livelli inferiori esistano o esisteranno, potremo scrivere il primo livello di DSLINE, un programma che disassembla una linea:

```
DSLINE    JSR LEGBYT     Legge il byte selezionato
          PHA            Lo salva nello stack
          JSR MNEMON     Stabilisce lo mnemonico corrispon-
                         dente
          JSR SPAZIO     Lascia uno spazio
          PLA            Rimette l'opcode nell'accumulatore
```

	JSR OPERND	Stabilisce l'operando richiesto
	JSR FINITO	Termina la linea visualizzando i campi da 3 a 6
	JSR NEXTSL	Seleziona il byte successivo
	RTS	

Come determinare lo mnemonico

Ci occorre una routine chiamata MNEMON, che determini lo mnemonico corrispondente all'opcode letto. Per poter ottenere questo risultato, sarà necessario avere disponibile in memoria una tabella di corrispondenza (come quella dell'Appendice A3). Una siffatta tabella però richiederebbe ben 768 byte di memoria, essendoci 256 possibili opcode ed essendo lo mnemonico composto da tre byte (256×3=768). Forse, organizzando in maniera più razionale i dati, occorrerà meno memoria.
Per esempio, perché includere lo stesso mnemonico più di una volta? Ci sono 8 differenti opcode che usano lo mnemonico LDA, quindi perché sprecare ben 24 byte di memoria per lo stesso mnemonico? Potremmo creare una tabella dei nomi, cioè nient'altro che una lista alfabetica degli mnemonici. Ci sono solamente 56 mnemonici e anche aggiungendone uno, ERR, per segnalare quando un opcode non è valido, ne avremo 57, il che vuol dire una tabella di 171 byte.
Come sarà possibile stabilire lo mnemonico corrispondente ad un dato opcode?
Tenendo presente che uno mnemonico è identificato da un unico numero, supponiamo di avere una tabella che dia, per ogni opcode, il corrispondente mnemonico. Sarà possibile, guardando la tabella dei codici mnemonici, trovare il numero che identifica l'opcode in questione e poi usare questo numero come indice nella tabella dei nomi. I tre caratteri consecutivi che si troveranno nella tabella dei nomi corrisponderanno al nome del vostro opcode.
Questo metodo richiede non una ma due tabelle. Comunque, le due tabelle richiedono molto meno spazio di memoria che non la tabella utilizzata nella prima ipotesi. La tabella dei codici sarà di 256 byte poiché dovrà prevedere spazio per ogni possibile opcode, anche per quelli eventualmente errati. La tabella dei nomi, invece, sarà di 171 byte. Quindi, le due tabelle occuperanno 427 byte, cioè il 43% in meno della prima tabella.

MNEMON	LDX #3	In uno mnemonico ci sono 3 lettere
	STX LETTER	
	TAX	Si prepara ad usare l'opcode come indice

```
          LDA MCODES,X    Cerca il codice mnemonico
                          dell'opcode (MCODES è la tabella
                          dei codici)
          TAX             Si prepara ad usare il codice tro-
                          vato come indice
MNLOOP    LDA MNOMI,X     Legge il nome dello mnemonico
                          corrispondente
          STX TEMP.X
          JSR PR.CAR      Stampa il carattere
          LDX TEMP.X
          INX             Legge la lettera successiva
          DEC LETTER      Se non è la terza lettera del nome
          BNE MNLOOP      rientra nel loop
          RTS
TEMP.X    .BYTE 0
LETTER    .BYTE 0
```

Come noterete la routine richiede solamente 30 byte per il programma, 2 byte per le variabili e 427 byte per le due tabelle (MNOMI e MCODES). L'intera subroutine quindi richiederà 459 byte ma, siccome la maggior parte di questi è assegnata alle tabelle di dati, ci saranno meno possibilità di errori. In caso di errori nell'inserimento delle tabelle, probabilmente il disassemblatore stamperà caratteri inesatti, ma la routine non si fermerà.
Dati errati comporteranno ovviamente interpretazioni errate, ma almeno la routine funzionerà e una routine funzionante è molto più facile da correggere di una che al minimo errore fa cadere il sistema.
Le tabelle richieste sono contenute nell'Appendice C8.

Come stampare l'operando

Siamo così arrivati alla parte più complicata del disassemblatore: la stampa dell'operando di un opcode. Disassemblando manualmente si compiono due passi: prima si determina il modo di indirizzamento dell'opcode, e poi, in base a questo, si scrive l'operando, ammesso che sia richiesto, nella giusta forma.
Per giusta forma si intende una virgola con X e Y per ogni istruzione indicizzata, oppure le parentesi per quelle indirette e, inoltre, scrivere tutti gli indirizzi con il byte alto in prima posizione, perché ciò rende più leggibile il programma.
OPERND sarà una subroutine che scrive l'operando di un dato opcode

determinando il modo di indirizzamento e poi chiamando un'altra subroutine che si occupi di quel modo.

OPERND

OPERND	TAX	Cerca il modo di indirizzamento
	LDA MODI,X	dell'opcode
	TAX	Adesso X indica il modo
	JSR MODO.X	Chiama la subroutine dei modi
	RTS	

MODI è una tabella che contiene il modo di indirizzamento degli opcode. Noterete che questa routine ne richiama un'altra, MODO.X, che si occupa del modo di indirizzamento "X". Come può avvenire ciò? Una maniera è quella di avere una tabella di puntatori in cui l'x-esimo puntatore indica la subroutine che si occupa del modo "X". MODO.X dovrà in seguito passare il controllo all'x-esima subroutine della tabella. Sarebbe comodo se il 6502 avesse una istruzione JSR indicizzata che permetterebbe di chiamare la subroutine il cui indirizzo è l'x-esimo della tabella. Sfortunatamente però non è prevista questa istruzione e quindi dovremo simularla con il software.
Esiste una istruzione di JMP indiretto. Se un puntatore, che chiameremo SUBPTR, indicherà una data subroutine, allora l'istruzione JMP (SUBPTR) trasferirà il controllo alla subroutine specificata. Quindi, MODO.X dovrà solo eguagliare SUBPTR all'x-esimo puntatore della tabella dei puntatori e, con l'istruzione JMP (SUBPTR), trasferire il controllo alla subroutine richiesta.

MODO.X

MODO.X	LDA SUBS,X	Legge il byte basso dell'x-esimo puntatore
	STA SUBPTR	
	INX	Incrementa l'indice per leggere il byte successivo
	LDA SUBS,X	
	STA SUBPTR+1	
	JMP (SUBPTR)	Trasferisce il controllo alla subroutine richiesta che non ritornerà qui ma al programma che ha richiamato MODO.X

SUBS	Tabella di puntatori, l'x-esimo dei quali punta alla subroutine che gestisce l'indirizzamento in modo X

Utility del disassemblatore

Avendo sviluppato MODO.X, la routine OPERND sarà in grado di chiamare la subroutine corrispondente al modo di indirizzamento. Adesso ci occorrono 13 differenti subroutine che si occupino dei 13 modi di indirizzamento del 6502.
Prima di scrivere queste subroutine, pensiamo a quello che dovranno compiere e vediamo se è possibile scrivere qualche utility per semplificare le funzioni richieste alle subroutine chiamando quelle del caso e nel giusto ordine.
La lista seguente riporta le utility che saranno necessarie:

ONEBYT	Stampa operando di 1 byte
TWOBYT	Stampa operando di 2 byte
RPAREN	Stampa una parentesi chiusa
LPAREN	Stampa una parentesi aperta
XINDEX	Stampa una virgola e X
YINDEX	Stampa una virgola e Y

ONEBYT

```
ONEBYT    JSR INC.SL      Avanza all'opcode successivo
          JSR DUMPSL      Lo stampa in esadecimale
          RTS
```

TWOBYT

Un operando di due byte specifica sempre un indirizzo con il byte basso prima. Per stampare l'operando con in prima posizione il byte alto, sarà necessario prima scrivere il secondo e poi il primo byte.

```
TWOBYT    JSR INC.SL      Avanza al primo byte dell'operando
          LDA LEGBYT      Lo mette nell'accumulatore
          PHA             Lo salva
          JSR INC.SL      Legge il secondo byte
          JSR DUMPSL      Lo stampa in esadecimale
```

```
            PLA             Richiama il primo byte
            JSR PR.BYT      Lo stampa in esadecimale
            RTS
```

Sia ONEBYT che TWOBYT lasciano il puntatore SELEZ indicante l'ultimo byte dell'operando.

RPAREN-LPAREN

RPAREN stampa una parentesi chiusa e LPAREN una parentesi aperta.

```
RPAREN      LDA #')         Carica A con il codice ASCII ")"
            BNE MANDA       Lo scrive
LPAREN      LDA #'(         Carica A con il codice ASCII "("
MANDA       JSR PR.CAR      Subroutine di stampa
            RTS
```

XINDEX

XINDEX stampa una virgola seguita da X.

```
XINDEX      LDA #',         Carica A con il codice ASCII di ","
            JSR PR.CAR
            LDA #'X         Carica A con il codice ASCII di "X"
            JSR PR.CAR
            RTS
```

YINDEX

YINDEX stampa una virgola seguita da Y.

```
YINDEX      LDA #',         Carica A con il codice ASCII di ","
            JSR PR.CAR
            LDA #'Y         Carica A con il codice ASCII di "Y"
            JSR PR.CAR
            RTS
```

Queste sono le utility che ci occorrono per il disassemblatore. A questo punto non ci resta che scrivere le routine per i modi di indirizzamento.

Subroutine di indirizzamento

Poiché il 6502 ha 13 modi differenti di indirizzamento, ci occorreranno 13 routine:

Subroutine	**Modo**
ABSLUT	Assoluto
ABS.X	Assoluto,X
ABS.Y	Assoluto,Y
ACC	Accumulatore
IMPLID	Implicito
IMMEDT	Immediato
INDRCT	Indiretto
IND.X	Indiretto,X
IND.Y	Indiretto,Y
RELATV	Relativo
ZEROPG	Pagina zero
ZERO.X	Pagina zero,X
ZERO.Y	Pagina zero,Y

Il compito principale di tutte queste subroutine è quello di stampare l'operando nella forma corretta. Benché un determinato modo di indirizzamento avrà sempre lo stesso numero di caratteri nell'operando, differenti modi di indirizzamento avranno operandi di differente lunghezza. Per esempio, l'indirizzamento implicito non ha caratteri nell'operando mentre l'indirizzamento indicizzato indiretto ne richiede ben sei.
Ma non ha importanza il numero di caratteri che compone l'operando, è necessario essere sicuri che il campo 3, quello degli indirizzi, cominci sempre alla stessa colonna. Pertanto, ogni subroutine di indirizzamento tornerà con il registro A che contiene il numero di caràtteri dell'operando, il registro X che contiene il numero di byte dell'operando e con il puntatore SELEZ che indica l'ultimo byte dell'operando (o dell'opcode se si tratta di un'istruzione da 1 byte). A questo punto la routine FINITO sarà in grado di lasciare l'appropriato numero di spazi prima di stampare i campi da 3 a 6.

Modo assoluto: ABSLUT

Per stampare l'operando di un indirizzamento assoluto ci occorre solo l'utility che scrive l'operando di 2 byte. Perciò, 8D B2 04 sarà disassemblato come:

```
STA 04B2   8D B2 04

ABSLUT     JSR TWOBYT
           LDX #2          X contiene il numero di byte
                           dell'operando
           LDA #4          A contiene il numero di caratteri
                           dell'operando
           RTS
```

Modo assoluto,X: ABS.X

Per stampare l'operando di un indirizzamento assoluto, X, dovremo stampare i due byte seguiti da ",X"

```
LDA D09A,X   BD 9A D0

ABS.X      JSR ABSLUT      Stampa l'operando di 2 byte
           JSR XINDEX      Stampa la virgola seguita da X
           LDX #2          X contiene il numero di byte
                           dell'operando
           LDA #6          A contiene il numero di caratteri
                           dell'operando
           RTS
```

Modo assoluto,Y: ABS.Y

In questo caso è necessario stampare l'operando di 2 byte seguito da ",Y":

```
ORA 02FE,Y   19 FE 02

ABS.Y      JSR ABSLUT      Stampa l'operando di 2 byte
           JSR YINDEX      Stampa la virgola seguita da X
           LDX #2          X contiene il numero di byte
                           dell'operando
           LDA #6          A contiene il numero di caratteri
                           dell'operando
           RTS
```

Modo accumulatore: ACC

In questo modo è sufficiente stampare una "A":

ROR A 6A

ACC	LDA #'A	Carica l'accumulatore con l'ASCII "A"
	JSR PR.CAR	Chiama la routine di stampa
	LDX #0	X contiene il numero di byte dell'operando
	LDA #1	A contiene il numero di caratteri dell'operando
	RTS	

Modo implicito: IMPLID

Questo modo non ha operandi quindi...

CLC 18

IMPLID	LDX #0	X contiene il numero di byte dell'operando
	LDA #0	A contiene il numero di caratteri dell'operando
	RTS	

Modo immediato: IMMEDT

Questo modo richiede l'operando di un byte quindi "A9 41" verrà disassemblato come segue:

LDA #$41 A9 41

IMMEDT	LDA #'#	Stampa il simbolo #
	JSR PR.CAR	
	LDA #'$	Stampa il simbolo $
	JSR PR.CAR	
	JSR ONEBYT	Stampa l'operando di un byte
	LDX #1	X contiene il numero di byte dell'operando
	LDA #4	A contiene il numero di caratteri dell'operando
	RTS	

Modo indiretto: INDRCT

In questo modo basterà stampare l'operando assoluto tra parentesi. Quindi i tre byte consecutivi "6C 00 04" saranno disassemblati come segue:

```
JMP (0400)   6C 00 04

INDRCT       JSR LPAREN      Stampa la parentesi aperta
             JSR ABSLUT      Stampa l'operando di 2 byte
             JSR RPAREN      Stampa la parentesi chiusa
             LDX #2          X contiene il numero di byte
                             dell'operando
             LDA #6          A contiene il numero di caratteri
                             dell'operando
             RTS
```

Modo indiretto,X: IND.X

Per questo modo è necessario stampare una parentesi aperta, un indirizzo di pagina zero, una virgola seguita da X e una parentesi chiusa. I due byte consecutivi "A1 3C" saranno così disassemblati:

```
LDA (3C,X)   A1 3C

IND.X        JSR LPAREN      Stampa una parentesi aperta
             JSR ZERO.X      Stampa l'indirizzo di pagina zero,
                             una virgola e X
             JSR RPAREN      Stampa una parentesi chiusa
             LDX #1
             LDA #6
             RTS
```

Modo indiretto,Y: IND.Y

In questo modo dovremo stampare una parentesi aperta, un indirizzo di pagina zero, una parentesi chiusa e una virgola seguita da Y. Quindi disassemblando "B1 AF" otterremo:

```
LDA (AF),Y   B1 AF

IND.Y        JSR LPAREN      Stampa una parentesi aperta
             JSR ZEROPG      Stampa l'indirizzo di pagina zero
```

```
         JSR RPAREN      Stampa una parentesi chiusa
         JSR YINDEX      Stampa una virgola e Y
         LDX #1
         LDA #6
         RTS
```

Modo relativo: RELATV

Il modo relativo è abbastanza anomalo. Un'istruzione di salto relativo può specificare un salto in avanti se l'operando è *più* (nell'intervallo 00-$7F) e indietro se l'operando è *meno* (nell'intervallo $80-$FF). Pertanto, per determinare l'indirizzo specificato dall'istruzione di salto relativo, dobbiamo prima di tutto determinare se l'operando è più o meno in modo da sapere se si salta avanti o indietro. Eseguita questa operazione è necessario sommare o sottrarre i 7 LSB dell'operando all'indirizzo immediatamente seguente l'operando stesso. Il risultato sarà l'indirizzo specificato dall'istruzione di salto relativo.

```
RELATV   JSR INC.SL      Seleziona il byte successivo
         JSR SAL.SL      Salva SELEZ nello stack
         JSR LEGBYT      Legge il byte dell'operando
         PHA             Lo salva nello stack
         JSR INC.SL      Incrementa SELEZ in modo che
                         punti al byte immediatamente suc-
                         cessivo all'istruzione di salto relativo
         PLA             Recupera il byte operando dallo
                         stack
         CMP #0          È più o meno?
         BPL AVANTI      Se è più, significa salto in avanti
         DEC SELEZ+1     Un salto all'indietro è uguale a un
                         salto in avanti eseguito da una lo-
                         cazione 256 byte più bassa
AVANTI   CLC             Somma il byte operando all'indirizzo
         ADC SELEZ       che segue l'istruzione di salto
         BCC RELEND
         INC SELEZ+1
RELEND   STA SELEZ       Adesso SELEZ punta all'indirizzo
                         specificato dall'istruzione di salto
                         relativo
         JSR PR.ADR
         JSR REC.SL      Resetta il puntatore SELEZ
         LDX #1
         LDA #4
         RTS
```

Modo pagina zero: ZEROPG

In questo modo dobbiamo stampare solo l'operando di un byte, quindi i due byte "85 2A" saranno disassemblati come segue:

```
STA 2A   85 2A

ZEROPG   LDA #0          Stampa due caratteri ASCII "0"
         JSR PR.BYT
         JSR ONEBYT      Stampa l'operando di un byte
         LDX #1
         LDA #2
         RTS
```

Modo indicizzato in pagina zero: ZERO.X e ZERO.Y

In questo modo dovremo stampare solamente l'indirizzo di pagina zero e una virgola seguita da X o Y. Pertanto "B5 6C" sarà disassemblato come:

```
LDA 6C,X   B5 6C
```

e "B6 53" come:

```
LDX 53,Y   B6 53

ZERO.X   JSR ZEROPG      Stampa l'indirizzo di pagina zero
         JSR XINDEX      Stampa una virgola e una X
         LDX #1
         LDA #2
         RTS
ZERO.Y   JSR ZEROPG
         JSR YINDEX      Stampa una virgola e una Y
         LDX #1
         LDA #2
         RTS
```

Come gestire i testi

A questo punto abbiamo tutte le subroutine che ci occorrono per disassemblare il codice macchina del 6502 nei 13 modi di indirizzamento riconosciuti.
Ma come faremo nel caso di testi o di messaggi inseriti in codice macchi-

na? Sappiamo che il nostro programma include alcuni messaggi che iniziano con TEX ($7F) e finiscono con ETX ($FF).
Il disassemblatore, al momento, non è al corrente di questo fatto e se cerchiamo di disassemblare dei programmi che contengono stringhe di testo, interpreterà queste stringhe come operandi e non fornirà l'esatta rappresentazione del codice letto.
Non sarebbe comodo disporre di una routine che riconosca il carattere TEX e stampi quello che segue come testo piuttosto che come opcode e operando? Finita la stringa il disassemblatore dovrebbe riconoscere anche il carattere ETX e riprendere a listare in modo convenzionale.
Questa routine non è difficile da implementare. Dobbiamo solamente definire uno pseudo-modo, che chiameremo TEXT e fare in modo che il carattere TEX sia il solo opcode che richiede il modo TEXT. Quindi scriveremo una routine per il modo TEXT, TXMODE, che stamperà come testo gli operandi compresi tra il carattere TEX e ETX.

TXMODE

```
TXMODE   PLA              Toglie l'indirizzo di ritorno
         PLA              a OPERND
         PLA              Toglie l'indirizzo di ritorno
         PLA              a DSLINE
TXLOOP   JSR NEXTSL       Prosegue dopo il pseudo opcode
                          TEX
         BMI TXFINE
         JSR LEGBYT       Legge il carattere
         CMP #ETX         È il carattere di fine (ETX)?
         BEQ TXFINE       Se sì, esce dal loop
         JSR PR.CAR       Se no, lo stampa
         CLC
         BCC TXLOOP       Ripete la sequenza
TXFINE   JSR CR.LF        Lascia una riga
         JSR NEXTSL
         RTS
```

Ora che abbiamo tutte le subroutine che ci occorrono possiamo costruire la tabella della subroutine di indirizzamento:

```
SUBS     .WORD ABSLUT
         .WORD ABS.X
         .WORD ABS.Y
         .WORD ACC
         .WORD IMPLID
```

```
        .WORD IMMEDT
        .WORD INDRCT
        .WORD IND.X
        .WORD IND.Y
        .WORD RELATV
        .WORD ZEROPG
        .WORD ZERO.X
        .WORD ZERO.Y
```

Ogni subroutine di indirizzamento tornerà dopo l'esecuzione con il puntatore SELEZ che indica l'ultimo byte dell'istruzione, con A che contiene il numero dei caratteri dell'operando e con X che contiene il numero di byte dell'operando.

La fine della linea: FINITO

La routine FINITO dovrà lasciare l'appropriato numero di spazi, stampare l'indirizzo dell'opcode ed effettuare un dump di 1, 2 o 3 byte a seconda dei casi.
Inoltre dovrà posizionare la stampante sulla riga successiva e incrementare il puntatore SELEZ al primo byte seguente la linea disassemblata e fino a che il valore di SELEZ è diverso da EA, cioè l'indirizzo di fine. FINITO tornerà con più se ci sono altri byte da disassemblare e con meno se ha raggiunto EA.

```
FINITO   STA OPCAR      Salva la lunghezza dell'operando
         STA OPBYT      in caratteri e in byte
         DEX            Se necessario, decrementa
         BMI SEL.OK     SELEZ perché punti
LOOP.1   JSR DEC.SL     l'opcode
         DEX
         BPL LOOP.1
SEL.OK   SEC            Lascia gli spazi necessari
         LDA INDCOL
         SBC #4         L'operando partiva alla colonna 4
         SBC OPCAR      e include OPCAR caratteri
         TAX            Servono quindi X spazi
         JSR SPAZI
         JSR PR.ADR     Stampa l'indirizzo dell'opcode
         JSR SPAZIO
LOOP.2   JSR DUMPSL     Stampa il byte selezionato
         JSR INC.SL
```

```
         DEC OPBYT        È l'ultimo byte dell'istruzione?
         BPL LOOP.2       Se no torna al loop
         JSR DEC.SL       Riporta SELEZ all'ultimo byte
                          dell'operando
FINEND   JSR CR.LF
         RTS
OPBYT    .BYTE            Numero di byte dell'operando
OPCAR    .BYTE 0          Numero di caratteri dell'operando
INDCOL   .BYTE 16         Colonna di partenza del campo in-
                          dirizzi
```

Adesso siamo in grado di disassemblare una linea. Scriviamo quindi le routine per disassemblare su video o su stampante. Queste due routine avranno la stessa struttura di TVDUMP e PRDUMP.

Disassemblare sullo schermo: TV.DIS

```
TV.DIS   LDA DISLNX       Inizializza il contatore di linea
         STA LINUM        con il numero di linee da disassem-
                          blare
         LDA #$FF         Setta l'indirizzo di fine a $FF
         STA EA
         STA EA+1
         JSR TVT.ON       Attiva il video
TVLOOP   JSR DSLINE       Disassembla una linea
         DEC LINUM
         BNE TVLOOP       Se non era l'ultima prosegue
         RTS
DISLNX   .BYTE 5          Stabilisce il numero di linee da
                          disassemblare; per disassemblarne
                          una alla volta settare DISLNX=1
LINUM    .BYTE 0          Questa variabile tiene conto delle
                          linee disassemblate
```

Disassemblare sulla stampante: PR.DIS

La routine visualizzerà il messaggio "DISASSEMBLER SU STAMPANTE" sullo schermo ma non sulla stampante. Quindi permetterà all'utente di definire l'indirizzo di partenza e quello di arrivo come per PRDUMP. Una volta specificato il blocco di memoria da disassemblare la routine stamperà il disassemblato sulla stampante e sul video.

```
PR.DIS    JSR PR.OFF       Disattiva la stampante
          JSR TVT.ON
          JSR INVIA:       Visualizza il titolo
          .BYTE TEX
          .BYTE CR,LF
          .BYTE            'DISASSEMBLER SU STAMPAN-
                           TE'
          .BYTE CR,LF,ETX
          JSR SETADS       Legge gli indirizzi di partenza e
                           fine specificati all'utente
          JSR GOTOSA       Setta SELEZ all'indirizzo di par-
                           tenza
          JSR PR.ON        Attiva la stampante
PRLOOP    JSR DSLINE       Disassembla una linea
          BPL PRLOOP       Se non era l'ultima prosegue con
                           la successiva
          RTS
```

Con PR.DIS e TV.DIS sarà possibile disassemblare qualsiasi blocco di memoria inviando il listato al video o alla stampante.

Capitolo

Utility di trasferimento 10

Molti programmi impiegano una grande quantità di tempo per trasferire dati da un'area di memoria all'altra. Sarebbe quindi molto comodo richiamare una utility di trasferimento per eseguire questi lavori.
Una utility di trasferimento dovrebbe:

— Essere abbastanza generalizzata per poter trasferire qualsiasi blocco di qualsiasi dimensione da una locazione di memoria ad un'altra
— Non bloccarsi se il blocco di partenza si sovrappone all'area di destinazione
— Avere punti di ingresso e configurazione variabili per poter soddisfare le esigenze dei differenti programmi che ne fanno uso
— Mantenere inalterati i puntatori
— Essere veloce

Questa routine sarà utilizzata molto spesso e risiederà in meno di 200 byte, meno del 3% della memoria disponibile su un sistema con 8K RAM.

Configurazione

Differenti programmi potranno trovare comodo poter disporre di differenti configurazioni dell'utility, per questo motivo cercheremo di preparare diverse configurazioni, ognuna con alcuni parametri da inizializzare. Le seguenti subroutine di ingresso dovrebbero soddisfare tutti i programmi che richiedono l'utility:

MOV.EA	Trasferisce un blocco, definito da un indirizzo di partenza (SA), da uno di fine (EA), all'indirizzo di destinazione
MOVNUM	Trasferisce un blocco, definito dal suo indirizzo di partenza, dal numero di byte del blocco (NUM), all'indirizzo di destinazione

MOV.EA fungerà semplicemente da routine d'input per MOVNUM. Setterà NUM=indirizzo di fine−indirizzo di partenza del blocco da muovere.

Sovrapposizioni

Non avremo problemi di sovrapposizione dei blocchi se trasferiremo i blocchi copiando verso l'alto quando si parte dal byte più alto e verso il basso quando si parte con il byte più basso del blocco di origine.
Per evitare le sovrapposizioni, MOVNUM deve determinare se sta copiando verso l'alto e verso il basso. Tuttavia, prima di iniziare il trasferimento, dovrà stabilire se l'indirizzo di destinazione è maggiore o minore dell'indirizzo di origine. Stabilito questo potrà richiamare le routine MOVSU o MOVGIU a seconda del caso.

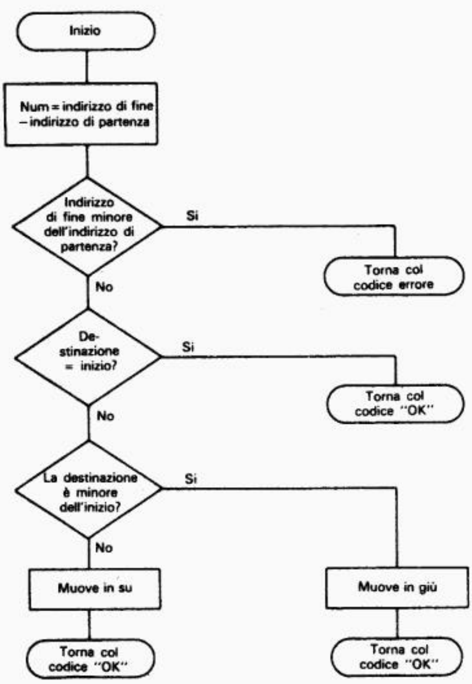


Figura 10.1 Diagramma di flusso del primo livello delle routine MOV.EA e MOVNUM

Usando il diagramma di flusso visibile in Figura 10.1 scriviamo una prima traccia delle routine MOV.EA e MOVNUM:

```
          LEGPTR=0             Puntatore della pagina di origine
          SCRPTR=LEGPTR+2      Puntatore della pagina di destinazione
MOV.EA    SEC                  Setta NUM=EA-SA
          LDX EA+1
          LDA EA
          SBC SA
          STA NUM
          BCS MOVE.1
          DEX
          SEC
MOVE.1    TXA
          SBC SA+1
          STA NUM+1
          BCS MOVNUM           NUM=EA-SA
ER.RTN    LDA #ERROR           Se EA<SA torna con codice errore
          RTS
MOVNUM    LDY #3               Salva i 4 byte di pagina zero che useremo
SAVE      LDA LEGPTR,Y
          PHA
          DEY
          BPL SAVE
          SEC                  Destinazione<origine?
          LDA SA+1
          CMP DEST+1
          BCC MOVSU
          BNE MOVGIU
          LDA SA               SA è nella stessa pagina?
          CMP DEST
          BCC MOVSU
          BNE MOVGIU
OK.RTN    LDY #0               Resetta i byte usati in pagina zero
RESET     PLA
          STA LEGPTR,Y
          INY
          CPY #4
          BNE RESET            Torna se sono stati resettati tutti e quattro
          RTS
```

```
NUM        .WORD 0        Variabile contenente il nume-
                          ro di byte da muovere
```

Ottimizzazione per ottenere la massima velocità

Trasferire una pagina di memoria per volta è la maniera più veloce di muovere i dati; per grossi blocchi di memoria possiamo trasferire la maggior parte dei byte in questo modo. Tuttavia, nel muovere dei dati, sposteremo una pagina di memoria per volta fino a che non resti meno di una pagina da spostare; allora muoveremo un byte alla volta fino a che tutto il blocco di memoria non sia stato trasferito. Le routine MOVSU e MOVGIU devono controllare se è rimasta più o meno di una pagina di

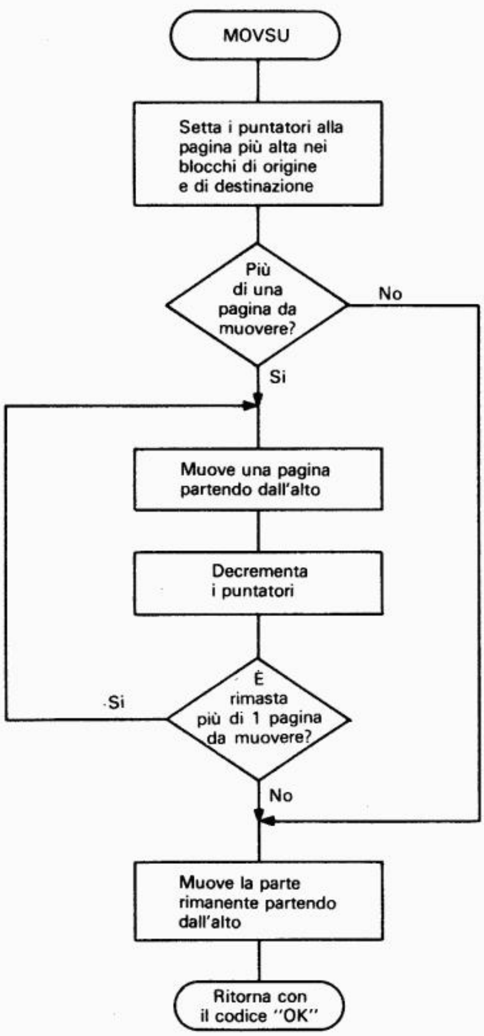


Figura 10.2 Diagramma di flusso della routine MOVSU

memoria da spostare e, in base a questo, saltare all'appropriata subroutine che permette di spostare una pagina o meno di una pagina.

MOVSU

Usando il diagramma di flusso visibile in Figura 10.2, possiamo scrivere il programma sorgente di MOVSU:

MOVSU	LDA NUM+1	C'è più di una pagina da muovere?
	BEQ MENOSU	Se no muove meno di una pagina
		Per muovere più di una pagina, setta i puntatori di pagina LEGPTR e SCRPTR alla pagina più alta del blocco di origine e di destinazione. Per ottenere questo bisogna usare X come byte alto e Y come byte basso di un puntatore che chiameremo (X,Y). Prima di tutto settiamo (X,Y)= NUM−$FF che è l'indirizzo relativo della pagina più alta del blocco
	LDY NUM+1	Mette in Y il byte alto della dimensione del blocco
	LDA NUM	Mette in A il byte basso della dimensione del blocco
	SEC	Prepara la sottrazione
	BSC #$FF	
	BCS NEXT.1	
	DEY	
NEXT.1	TAX	Ora (X,Y)=NUM−$FF X è il byte basso e Y quello alto di NUM−$FF
	STY SCRPTR+1	
	TXA	
	CLC	Prepara l'addizione
	ADC SA	
	STA LEGPTR	
	BCC NEXT.2	
	INY	

NEXT.2	TYA	
	ADC SA+1	
	STA LEGPTR+1	Ora LEGPTR=SA+NUM−$FF (l'ultima pagina del blocco di origine)
	TXA	
	CLC	Prepara l'addizione
	ADC DEST	
	STA SCRPTR	
	BCC NEXT.3	
	INC SCRPTR+1	
NEXT.3	LDA SCRPTR+1	
	ADC DEST+1	
	STA SCRPTR+1	A questo punto SCRPTR=DEST+NUM−$FF (l'ultima pagina nel blocco di destinazione). Quindi i puntatori LEGPTR e SCRPTR puntano rispettivamente all'ultima pagina del blocco di origine e destinazione
	LDX NUM+1	X=numero di pagine da muovere
PAGSU	LDY #$FF	Muove una pagina
SULOOP	LDA (LEGPTR),Y	Legge un byte dal blocco di origine
	STA (SCRPTR),Y	Scrive il byte nel blocco di destinazione
	DEY	Setta l'indice per il byte successivo
	BNE SULOOP	Esegue di nuovo se non era l'ultimo byte
	LDA (LEGPTR),Y	Muove l'ultimo byte
	STA (SCRPTR),Y	
	DEC SCRPTR+1	Decrementa il puntatore di
	DEC LEGPTR+1	pagina
	DEX	Ancora più di una pagina da muovere?
	BNE PAGSU	Se sì, muove un'altra pagina
MENOSU	JSR LOPAG	Setta LEGPTR e SCRPTR al fondo dei blocchi di origine e destinazione

```
            LDY NUM            Setta l'indice al numero di
                               byte da muovere
POCSU       LDA (LEGPTR),Y     Muove un byte
            STA (SCRPTR),Y
            DEY
            CPY #$FF
            BNE POCSU          Se non è l'ultimo ne muove
                               un altro
            JMP OK.RTN         Se sì, ritorna con il codice
                               "OK"
LOPAG       LDA SA             Setta i puntatori di pagina al
            STA LEGPTR         basso dei blocchi di origine e
            LDA SA+1           destinazione
            STA LEGPTR+1
            LDA DEST
            STA SCRPTR
            LDA DEST+1
            STA SCRPTR+1
            RTS
```

MOVGIU

La Figura 10.3 mostra un algoritmo che permette di muovere un blocco di dati verso il basso della memoria.
Usando la Figura 10.3 come guida, possiamo scrivere il programma sorgente di MOVGIU:

```
MOVGIU      JSR LOPAG          Setta i puntatori al basso dei bloc-
                               chi di origine e destinazione
            LDY #0             Y deve essere uguale a 0 in ogni
                               caso
            LDX NUM+1          C'è piu di una pagina da muovere?
            BEQ MENGIU         Se no, muove meno di una pagina
PAGGIU      LDA (LEGPTR),Y     Legge un byte dal blocco di origine
            STA (SCRPTR),Y     e lo scrive nel blocco di destinazione
            INY                È l'ultimo byte della pagina?
            BNE PAGGIU
            INC LEGPTR+1       Incrementa i puntatori
            INC SCRPTR+1
            DEX                Ancora più di una pagina da muo-
                               vere?
```

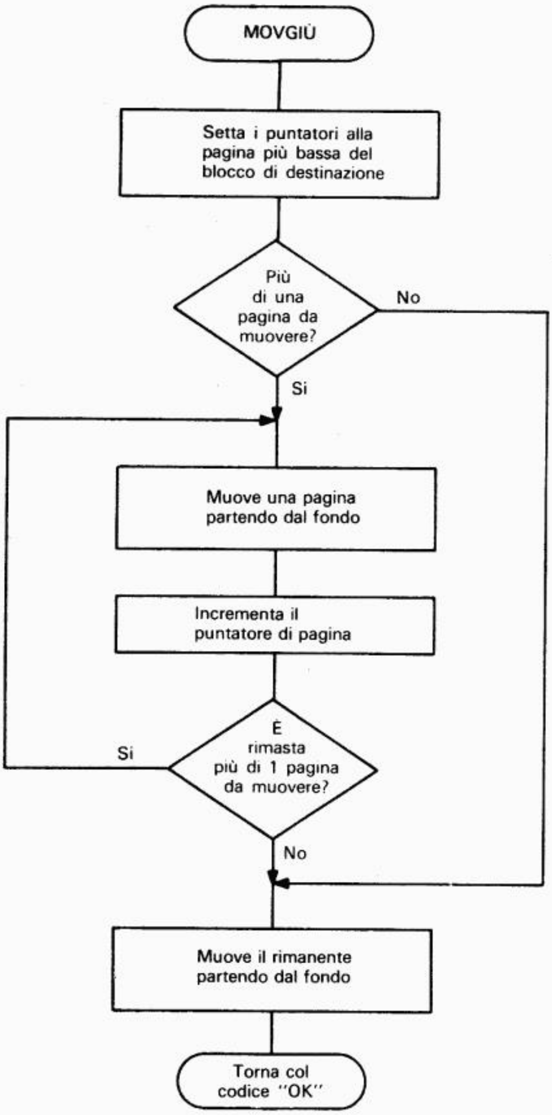


Figura 10.3 Diagramma di flusso della routine MOVGIU

```
          BNE PAGGIU        Se sì, muove un'altra pagina
          LDY #0            Muove meno di una pagina parten-
                            do dal basso
MENGIU    LDA (LEGPTR),Y    Legge un byte dal blocco di origine
          STA (SCRPTR),Y    e lo scrive nel blocco di destinazione
          INY               Setta l'indice per il byte successivo
          SEC
          CPY NUM           È l'ultimo byte?
          BNE MENGIU+1      Se no, ne muove un altro
          JMP OK.RTN        Se sì, torna col codice OK
```

Velocità

Per grossi blocchi di dati, la maggior parte dei byte verrà trasferita dalle subroutine PAGSU e PAGGIU. Poiché il 6502 spende la maggior parte del tempo in questi loop, vediamo quanto tempo ci vorrà per muovere un byte: le informazioni che ci servono sono contenute nell'Appendice A4. Generalmente non è il caso di entrare in particolari riguardanti la velocità di esecuzione di una piccola porzione di programma, ma questi due loop costituiscono il cuore dell'utility di trasferimento perché muovono la maggior parte dei byte che costituiscono il blocco. Rendendo questi due loop molto efficienti, sarà possibile rendere l'utility molto veloce. Infatti, questi loop permetteranno di muovere blocchi più grandi di una pagina ad una velocità che si avvicina a 16 cicli macchina per byte.
MOV.EA e MOVNUM sono utility perché hanno la possibilità di adattarsi a quasi tutti i programmi che le richiamano. Però non sono molto adeguate all'utente che desideri semplicemente muovere qualcosa. Con il Monitor Visibile e le utility di trasferimento è possibile muovere qualco-



Figura 10.4 Diagramma di flusso della routine MOVER

sa da una locazione a un'altra, ma è necessario conoscere gli indirizzi da settare e l'indirizzo dell'utility stessa.
Questo è troppo da ricordare; quello di cui abbiamo bisogno è uno strumento che conosca gli indirizzi e non ci richieda di ricordarli.
Programmando con il Monitor Visibile, potrebbe presentarsi la necessità di spostare alcuni dati o alcune linee di programma da una locazione a un'altra e sarebbe comodo poter richiamare una routine di trasferimento con la semplice pressione di un tasto, ad esempio M.
Supponiamo di premere il tasto M durante l'uso del Monitor Visibile: questo richiamerà l'utilità di trasferimento. La prima operazione compiuta dall'utility dovrebbe essere la segnalazione della sua attivazione. E se avessimo premuto il tasto M per errore? Il computer quindi deve segnalare che è stato richiamato un nuovo programma, visualizzando un titolo: "TRANSFER".
Poi dovrebbe permetterci di specificare gli indirizzi di inizio e fine e di destinazione di un blocco. Una volta specificati questi indirizzi, il programma potrà richiamare la routine MOV.EA che eseguirà il trasferimento in base ai dati forniti dall'utente.
Il primo livello di MOV.EA è molto semplice come si vede nel diagramma di flusso in Figura 10.4.

MOVER

MOVER	JSR TVT.ON	Attiva il video
	JSR INVIA:	Visualizza il titolo
	.BYTE TEX,CR	
	.BYTE 'TRANSFER'	
	.BYTE CR,LF,LF	
	.BYTE ETX	
	JSR SETADS	Legge gli indirizzi di inizio, fine e destinazione
	JSR SET.DA	
	JSR MOV.EA	Muove il blocco specificato dai puntatori
	RTS	Torna dopo aver mosso il blocco e con la pagina zero inalterata

Naturalmente la routine MOVER funzionerà solo se c'è una subroutine che permette all'utente di specificare l'indirizzo di destinazione.

L'indirizzo di destinazione: SET.DA

```
SET.DA    JSR TVT.ON          Attiva il video
          JSR INVIA:          Visualizza il titolo
          .BYTE TEX
          .BYTE CR,LF,LF
          .BYTE               "INDIRIZZO DI DESTINAZIONE"
          .BYTE               "(POI PREMERE Q)"
          .BYTE ETX
          JSR VISMON          Richiama il Monitor Visibile
                              perché l'utente possa specificare
                              l'indirizzo di partenza
DESQUI    LDA SELEZ
          STA DEST
          LDA SELEZ+1         Setta l'indirizzo di destinazione
                              uguale a quello specificato
                              dall'utente
          STA DEST+1
          RTS
DEST      .WORD 0
```

Nel Capitolo 12, è spiegato il metodo per agganciare l'utility di trasferimento al Monitor Visibile collegandola ad un dato tasto. Per muovere qualsiasi cosa in memoria sarà poi sufficiente premere il tasto stabilito e l'utility eseguirà quanto richiesto.

Capitolo

Un semplice text editor

11

Con il Monitor Visibile è possibile memorizzare dei testi posizionando la freccia indice sotto il secondo campo e digitando i caratteri necessari. Però con questo sistema è necessario l'uso di due tasti per ogni carattere del messaggio da inserire: prima il tasto corrispondente al carattere vero e proprio, poi lo spazio per incrementare l'indirizzo. Inoltre, se si volesse inserire uno spazio o un a capo bisognerebbe posizionare la freccia sotto il campo 1 e inserire l'equivalente in esadecimale, $20 per lo spazio e $0D per il CR. Quindi sarebbe necessario premere la barra spaziatrice per selezionare l'indirizzo successivo, il tasto "→" per poter inserire valori esadecimali e il tasto "←" per tornare sotto il campo dei caratteri.
Se il testo da inserire è limitato a una dozzina di caratteri, il Monitor Visibile può soddisfare le esigenze ma, dovendo memorizzare testi più lunghi sarebbe opportuno poter disporre di un sistema più comodo, ad esempio un sia pur semplice *text editor*.
Esistono text editor di diversa specie. Un text editor di linea, utile per creare i programmi sorgente, permetterà di creare o correggere una linea alla volta. Generalmente ogni linea è numerata, come in BASIC, quindi per poter modificare una linea sarà necessario richiamarla per numero. D'altra parte esistono text editor più potenti con cui è possibile ribattere, inserire o cancellare caratteri dovunque in una data stringa. Text editor di questo tipo sono incorporati generalmente nei word processor, ma non illudetevi, in questo capitolo presenteremo un software che non si avvicina nemmeno lontanamente alle prestazioni e alle sofisticazioni degli editor contenuti nel software dedicato alla gestione di testi. Tuttavia, presenteremo un text editor molto semplice che permetterà di modificare e inserire stringhe di caratteri ovunque in memoria.

Il text editor sarà strutturato nelle tre parti come mostrato in Figura 11.1:

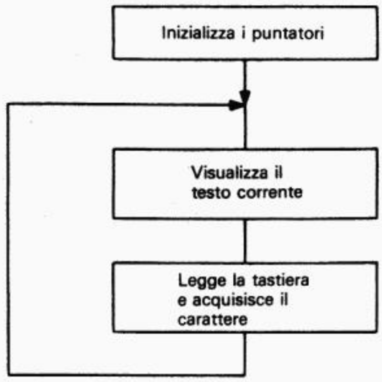


Figura 11.1 Struttura di un semplice text editor

Usando questo diagramma come guida possiamo scrivere il primo livello del programma sorgente.

EDITOR	JSR SETBUF	Inizializza le variabili e i puntatori necessari
EDLOOP	JSR MOSTRA	Visualizza una porzione del buffer
	JSR EDITA	Permette la correzione o il movimento all'interno del buffer
	CLC	
	BCC EDLOOP	Salta indietro per visualizzare il testo corrente

Sembra familiare? Certamente! Questa struttura è molto simile a quella del Monitor Visibile. Una struttura semplice e ben congegnata per soddisfare le esigenze dei programmi interattivi di visualizzazione.

SETBUF

L'editor lavorerà sui testi contenuti in una determinata porzione di memoria chiamata *buffer*. Poiché l'editor deve essere in grado di modificare il contenuto del buffer, quest'ultimo dovrà risiedere in un'area di RAM riservata esclusivamente allo scopo. A questo punto si annuncia un problema ben noto ai programmatori: come distribuire la memoria in maniera razionale.

La memoria usata per il programma non deve essere usata allo stesso tempo per i testi e similmente la memoria riservata ai testi non deve essere usata per il programma.

Come è possibile sistemare 5 chili di pomodori in un sacchetto che ne può contenere solo 4 senza strappare il sacchetto o schiacciare i pomodori? Non è possibile! Se volete memorizzare molte stringhe di testo potreste ritrovarvi senza più memoria disponibile per il text editor. D'altra parte un text editor sofisticato richiederebbe una gran quantità di memoria per se stesso e ne lascerebbe troppo poca disponibile alle stringhe di caratteri.
Il nostro text editor lascia all'utente la possibilità di stabilire la quantità di memoria da riservare al buffer. Una subroutine chiamata SETBUF, inizializza i puntatori all'inizio e alla fine del buffer di testo. Il resto dell'editor poi lavorerà esclusivamente sull'area di memoria definita da questi puntatori.
SETBUF quindi setta gli indirizzi di inizio e fine del buffer di testo. Se vorrete inserire ed usare sempre lo stesso buffer, sarà sufficiente usare una subroutine che specifichi gli indirizzi di inizio e fine che desiderate. In ogni altro caso sarà utile usare la versione di SETBUF che permette di stabilire le locazioni del buffer ogni volta che viene richiamata.
Per prova, si potrebbe usare la memoria riservata al video; questo permetterebbe di vedere esattamente cosa succede all'interno del buffer.

SETBUF

```
SETBUF   JSR TVT.ON                    Attiva il video
         JSR INVIA:                    Visualizza il titolo
         .BYTE TEX,CR,LF,LF
         .BYTE 'INIZIALIZZAZIONE BUFFER.'
         .BYTE CR,LF,LF,ETX
         JSR SETADS                    Legge gli indirizzi specifica-
                                       ti dall'utente
         JSR GOTOSA                    SELEZ=inizio del buffer
         RTS
```

Questa versione di SETBUF permette all'utente di allocare il buffer dovunque in memoria ammesso che l'indirizzo di fine non sia più basso in memoria di quello di partenza. All'uscita dalla subroutine il puntatore SELEZ indicherà l'indirizzo di partenza del buffer.

MOSTRA

Ora che SETBUF ha inizializzato i puntatori del buffer proviamo ad immaginare come potremmo visualizzare una porzione del buffer.
La Figura 11.2 mostra una semplice videata composta da tre linee.

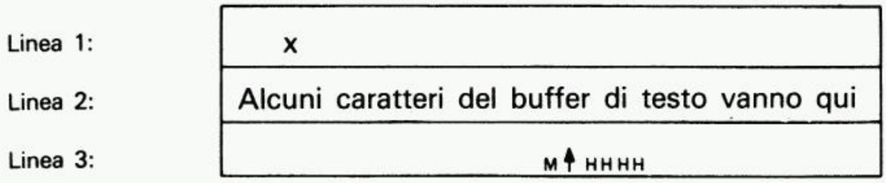


Figura 11.2 La videata del text editor

X segna la posizione "home". Tutto ciò che sarà visualizzato è relativo a questa posizione. Quindi, per muovere la videata dell'editor sullo schermo, basterà spostare la posizione home che viene specificata dalla routine MOSTRA.
La linea 1 è completamente vuota. Il suo scopo è quello di separare il testo visualizzato nella linea 2 da ciò che potrebbe essere presente sul video.
La linea 2 visualizza una stringa di caratteri contenuta nel buffer. Il carattere selezionato è indicato dalla freccia visibile nella linea 3 insieme al suo indirizzo rappresentato dai caratteri esadecimali HHHH.
La lettera M sempre nella linea 3 indicherà l'opzione selezionata per il text editor.

Modo sostituzione e modo inserimento

L'editor potrà funzionare in due modi: il modo *sostituzione* e il modo *inserimento*. In modo sostituzione sarà possibile ribattere o cambiare il carattere selezionato, mentre in modo inserimento sarà possibile aggiungere i caratteri che vengono immessi da tastiera. La possibilità di selezionare un modo piuttosto di un altro verrà trattata più avanti. Adesso bisogna fare in modo che la routine MOSTRA sappia quale modo è selezionato per poter visualizzare l'appropriato carattere nella linea 3 della videata dell'editor.
Poiché i modi possibili saranno solamente due, terremo traccia del modo selezionato per mezzo di una variabile da 1 byte che chiameremo EDMODO e a cui assegneremo i seguenti valori:

EDMODO=0 per il modo sostituzione
EDMODO=1 per il modo inserimento

Ogni altro valore di EDMODO non è definito e quindi illegale. Quindi, se la routine MOSTRA dovesse trovare EDMODO ad un valore illegale, dovrà essere in grado di settare questa variabile con un valore ammesso, diciamo 0. Ci occorrono poi due caratteri, SOSCAR e INSCAR, per visualiz-

zare il modo in cui si trova l'editor. Useremo la lettera S per il modo sostituzione e la lettera I per il modo inserimento.

MOSTRA

```
MOSTRA    JSR TVSALV     Salva i byte di pagina 0
          JSR TVHOME     Setta la posizione HOME
          LDX TVCOL      Pulisce tre linee
          LDY #3         del video
          JSR PULXY
          JSR TVHOME
          JSR TVGIU      Passa alla linea 2
          JSR TVSALV
          JSR LINE.2     Visualizza il testo della linea 2
          JSR TVREC
          JSR TVGIU      Passa alla linea 3
          JSR LINE.3     Visualizza il testo della linea 3
          JSR TVREC      Resetta i byte di pagina zero
          RTS
```

Naturalmente MOSTRA potrà funzionare solo se saranno presenti le routine LINE.2 e LINE.3 che visualizzeranno la seconda e la terza linea della videata.

Come visualizzare la linea di testo

Per visualizzare la linea di testo sarà sufficiente copiare un certo numero di caratteri dal buffer alla seconda linea dell'editor. Siccome il video ha tante colonne quante sono specificate da TVCOL dovremo copiare TVCOL caratteri dal buffer in modo che il carattere centrale sia effettivamente quello selezionato. Otterremo questo risultato dividendo TVCOL per due e sottraendo da SELEZ il valore ricevuto.

LINE.2

```
LINE.2    JSR SAL.SL     Salva il puntatore SELEZ
          LDA TVCOL
          LSR A
          TAX            X=TVCOL/2
          DEX
```

```
LOOP.1     JSR DEC.SL      Decrementa SELEZ X volte
           DEX
           BPL LOOP.1
           LDA TVCOL       Inizializza il contatore
           STA CONTAT      per visualizzare TVCOL caratteri
LOOP.2     JSR LEGBYT      Legge un carattere dal buffer
           JSR TV.SCR      Visualizza sullo schermo
           JSR TVVAI       Seleziona la posizione successiva
                           sul video
           JSR INC.SL      Seleziona il byte successivo del
                           buffer
           DEC CONTAT      È l'ultimo carattere del buffer?
           BPL LOOP.2      Se no, passa al successivo
           JSR REC.SL      Resetta SELEZ dallo stack
           RTS
```

Come visualizzare la linea di stato

La linea 3 dell'editor visualizza il modo selezionato, la freccia indice e l'indirizzo dell'attuale carattere.

LINE.3

```
LINE.3     LDA TVCOL
           LSR A           A=TVCOL/2
           SBC #2          A=(TVCOL/2)-2
           JSR TVPIU       TV.PTR adesso punta due caratteri
                           più a sinistra del centro della linea 3
           LDA EDMODO      Quale modo è selezionato?
           CMP #1          Il modo inserimento?
           BNE SOMODO      Se no, è il modo sostituzione
           LDA #INSCAR     Se sì, mette in A il carattere di in-
                           serimento
           CLC
           BCC TVMODO
SOMODO     LDA #SOSCAR     Mette in A il carattere di sostituzione
TVMODO     JSR TV.SCR      Visualizza il carattere di modo
           LDA #2
           JSR TVPIU       Ora TV.PTR punta al centro della
                           linea 3
           LDA FRECC       Visualizza la freccia indice
```

```
JSR TV.SCR
LDA #2
JSR TVPIU        Ora TV.PTR punta alla posizione ri-
                 servata all'indirizzo del carattere
                 corrente
LDA SELEZ+1      Visualizza l'indirizzo del carattere
JSR VEDBYT       corrente
LDA SELEZ
JSR VEDBYT
RTS
```

Abbiamo scelto di definire il corrente carattere selezionato dall'editor come il carattere a cui punta SELEZ. Abbiamo già sviluppato alcune subroutine per operare su SELEZ e sul byte selezionato, quindi non saremo costretti a scriverne di nuove, ma potremo usare la maggior parte delle utility di SELEZ presentate nei capitoli precedenti.

L'editor in pratica

Ora che siamo in grado di visualizzare le tre linee di informazioni necessarie analizziamo le funzioni dell'editor che sono più utili:

- La possibilità di spostarsi in avanti nel messaggio
- La possibilità di spostarsi all'indietro nel messaggio
- La possibilità di sostituire il carattere selezionato
- La possibilità di cancellare il carattere selezionato
- La possibilità di cancellare l'intero messaggio
- La possibilità di inserire caratteri alla posizione selezionata
- La possibilità di passare dal modo sostituzione al modo inserimento e viceversa
- La possibilità di visualizzare i messaggi
- La possibilità di uscire dall'editor

Quali tasti richiameranno tutte queste funzioni? Lascerò la possibilità di scelta all'utente, trattando i tasti come variabili e memorizzandoli in una tabella che chiameremo EDCOM (comandi editor - vedere Appendice C11). Quindi, per assegnare una determinata funzione ad un tasto sarà sufficiente memorizzare il codice del tasto scelto nell'appropriata locazione della tabella.

EDITA

```
EDITA    JSR LEGTAS     Legge il valore del tasto premuto
         CMP TFINE      È il tasto "fine"?
         BNE TFAI       Se no, esegue la funzione richiesta
         PHA            Salva il valore del tasto nello stack
                        Se l'utente ha digitato 2 volte il ta-
                        sto "fine" deve uscire dall'editor
         JSR LEGTAS
         CMP TFINE      È il tasto "fine"?
         BNE NOFINE     Se no, deve compiere le funzioni ri-
                        chieste, rispettando l'ordine dei ta-
                        sti
FINEED   PLA            Uscita dall'editor: leva dallo stack
         PLA            il valore del tasto "fine". Legge
         PLA            l'indirizzo di ritorno al primo livello
         RTS
NOFINE   STA TEMPCA     Salva il tasto che ha seguito quello
                        di "fine"
         PLA            Legge dallo stack il primo tasto "fi-
                        ne"
         JSR TFAI       Esegue la funzione richiesta
         LDA TEMPCA     Riporta nell'accumulatore il secon-
                        do tasto
TFAI     CMP TMODO      È il tasto di cambio modo?
         BNE IFSUCC     Se no, esegue il test successivo
         DEC EDMODO     Se sì, cambia modo
         BPL FINEFA
         LDA #1
         STA EDMODO
FINEFA   RTS
IFSUCC   CMP TSUCC      È il tasto "successivo"?
         BNE IFPREC     Se no, esegue il test successivo
         JSR NEXTCA     Se sì, il cursore avanza di una posi-
                        zione
         RTS
IFPREC   CMP TPREC      È il tasto "precedente"?
         BNE IFCANC     Se no, esegue il test successivo
         JSR PRECCA
         RTS
IFCANC   CMP TCANC      Tasto "cancellazione"? (carattere)
         BNE IFPRT
         JSR CANCEL
         RTS
```

```
IFPRT    CMP TPRT      Tasto "stampa"?
         BNE IFELIM
         JSR PRTBUF
         RTS
IFELIM   CMP TELIM     Tasto "elimina"? (testo)
         BNE TCAR
         JSR ELIM
         RTS
TCAR     LDX EDMODO    È in modo sostituzione?
         BEQ RIBAT     Se sì, sostituisce il carattere
         JSR INSER     Se no, inserisce
         RTS
RIBAT    JSR MEM.SL    Mette il carattere nella corrente
                       locazione
         JSR NEXTSL
         RTS
INSER    PHA           Salva il carattere da inserire
         JSR SAL.SL    Salva l'indirizzo del carattere sele-
                       zionato
         LDA SA+1      Salva la locazione di partenza del
                       buffer
         PHA
         LDA SA
         PHA
         LDA EA+1      Salva la locazione di fine del buffer
         PHA
         LDA EA
         PHA
         JSR SAQUI     SA=SELEZ
         JSR NEXTSL    Avanza alla successiva posizione
                       del buffer
         BMI FINEIN    Se fine buffer sostituisce
         JSR DESQUI    DEST=SELEZ
         LDA EA        Decrementa l'indirizzo di fine
         BNE NEXT      per muovere il testo oltre la fine
         DEC EA+1      del buffer
NEXT     DEC EA        Ora l'indirizzo di inizio è il caratte-
                       re corrente, quello di destinazione
                       è il carattere seguente e l'indirizzo
                       di fine è un carattere oltre l'ultimo
                       carattere del buffer. Quindi può
                       muovere un blocco
UNOSU    JSR MOV.EA    Crea spazio per un byte alla cor-
```

```
                                rente locazione muovendo a DEST
                                il blocco specificato da EA e SA
FINEIN     PLA                  Resetta EA
           STA EA
           PLA
           STA EA+1
           PLA                  Resetta SA
           STA SA
           PLA
           STA SA+1
           JSR REC.SL           Resetta SELEZ
           PLA                  Resetta l'accumulatore con il carat-
                                tere da inserire
           JSR RIBAT
           RTS
```

In questo modo, EDITA sarà in grado di compiere le funzioni richieste, ammesso che si scrivano le seguenti subroutine:

- — NEXTCA Seleziona il carattere successivo
- — PRECCA Seleziona il carattere precedente
- — ELIM Cancella il buffer
- — PRTBUF Stampa il contenuto del buffer
- — CANCEL Cancella un carattere

Seleziona il carattere successivo: NEXTCA

È necessario poter avanzare di posizione nell'ambito del buffer senza però andare oltre la fine del buffer o del messaggio. La fine del messaggio sarà indicata da uno o più ETX e, se il messaggio è più corto del buffer specificato, i caratteri mancanti per arrivare alla fine del buffer saranno tutti ETX. Quindi, se il carattere selezionato è un ETX, oppure se l'indirizzo specificato è la fine del buffer, non sarà possibile avanzare in memoria. Ma se non siamo alla fine del testo o alla fine del buffer, per selezionare il carattere successivo ci occorre una subroutine che chiameremo NEXTCA.

```
NEXTCA     JSR LEGBYT           Legge il carattere selezionato
           CMP #ETX             È ETX?
           BEQ AN.ETX           Se sì, ritorna con codice "MENO"
           JSR NEXTSL           Se no, seleziona il byte successivo
           RTS                  del buffer e ritorna con codice
                                "PIÙ"
AN.ETX     LDA #$FF
           RTS
```

Seleziona il carattere precedente: PRECCA

Questa subroutine funziona in maniera molto simile alla precedente. NEXTCA incrementa SELEZ e ritorna "PIÙ" a meno che SELEZ non sia maggiore o uguale a EA, nel qual caso NEXTCA conserva SELEZ e ritorna "MENO"; PRECCA invece decrementa SELEZ e ritorna "PIÙ" a meno che SELEZ non sia minore o uguale a SA, nel qual caso dovrebbe conservare SELEZ e ritornare "MENO".

```
PRECCA   SEC            Prepara la comparazione
         LDA SA+1       SELEZ>SA?
         CMP SELEZ+1
         BCC SL.OK      Se sì, SELEZ può essere decremen-
                        tato
         BNE NO.OK
         LDA SA
         CMP SELEZ
         BEQ NO.DEC     Se SELEZ=SA non decrementa
         BNE NO.OK
SL.OK    JSR DEC.SL     Se SELEZ è ok allora decrementa
         LDA #0         Ritorna con codice "PIÙ"
         RTS
NO.OK    LDA SA         Se SELEZ<SA lo porta a un valore
         STA SELEZ      accettabile SELEZ=SA
         LDA SA+1
         STA SELEZ+1
         LDA #0
         RTS
NO.DEC   LDA #$FF       SELEZ=SA quindi non decrementa
         RTS            e ritorna con codice "MENO"
```

Cancella il buffer: ELIM

Per cancellare il buffer sarà sufficiente riempirlo con ETX:

```
ELIM     JSR GOTOSA     SELEZ=prima posizione del buffer
ELOOP    LDA #ETX
         JSR MEM.SL     Mette ETX nella posizione selezio-
                        nata
         JSR NEXTSL     Seleziona la posizione successiva
         BPL ELOOP      Se non è l'ultima posizione ripete
         JSR GOTOSA     Se ha finito resetta SELEZ all'inizio
         RTS
```

Stampa il contenuto del buffer: PRTBUF

Per stampare il buffer dobbiamo stampare i caratteri contenuti fino al primo ETX e, se l'ETX non è contenuto nel buffer, non dovremo stampare i caratteri che si trovano oltre l'indirizzo di fine.

```
PRTBUF   JSR GOTOSA    SELEZ punta all'inizio del buffer
PRLOOP   JSR LEGBYT    Legge il carattere selezionato
         CMP #ETX      È ETX?
         BEQ FINEPR    Se sì, finisce di stampare
         JSR INVCAR    Se no, invia il carattere alle perife-
                       riche selezionate
         JSR NEXTSL    Seleziona il carattere successivo
         BPL PRLOOP    Se non è fine buffer prosegue con
                       il successivo
FINEPR   RTS
```

Cancella un carattere: CANCEL

Per cancellare il carattere selezionato basterà spostare a sinistra di un byte tutti i caratteri che seguono.

```
CANCEL   JSR SAL.SL    Salva l'indirizzo del carattere sele-
                       zionato
         LDA SA+1      Salva l'indirizzo di inizio del buffer
         PHA
         LDA SA
         PHA
         JSR DESQUI    Setta DEST=SELEZ perché deve
                       muovere un blocco di testo
         JSR NEXTSL    Avanza di un carattere nel buffer,
                       se possibile
         JSR SAQUI     Setta SA=SELEZ perché il blocco
                       da muovere parte 1 byte sopra il
                       carattere selezionato
         JSR MOV.EA    Muove il blocco specificato da SA,
                       EA e DEST
         PLA           Resetta SA
         STA SA
         PLA
         STA SA+1
         JSR REC.SL    Resetta SELEZ
         RTS
```

Questa è l'ultima subroutine di cui avevamo bisogno. A questo punto abbiamo abbastanza routine da combinare per ottenere un semplice text editor. Le Appendici C10 e C11 riportano i listati di questo text editor e mostrano quali tasti funzioneranno sul C-64 e sul VIC-20.
Se preferite definire i tasti funzione differentemente, è sufficiente cambiare il valore delle variabili nella tabella dei tasti. Se volete eliminare una funzione, memorizzate il valore zero nell'appropriata locazione della tabella.
Troverete questo text editor molto comodo per inserire tabelle di caratteri ASCII in memoria e per arricchire i vostri programmi con messaggi e commenti.

Capitolo 12
Il Monitor Visibile integrato

Giunti a questo punto abbiamo a disposizione il Monitor Visibile, le utility di stampa, due strumenti di dump, un disassemblatore, una utility di trasferimento e un semplice text editor. Non sarebbe comodo poter combinare il tutto in un unico pacchetto interattivo? Questo potrebbe permetterci di richiamare una qualsiasi funzione o utility con la semplice pressione di un tasto.
Poiché il Monitor Visibile fa già uso di diversi tasti (da A a F, da 0 a 9, G, RETURN, la barra spaziatrice, →, ← e CLEAR SCREEN) dovremo assegnare queste nuove funzioni a tasti non ancora utilizzati.
Di seguito è riportata una lista dei tasti e delle rispettive funzioni.

H	Richiama TVDUMP o, se la stampante non è attiva, PRDUMP
M	Richiama l'utility di trasferimento
P	Attiva il flag della stampante
T	Richiama il text editor
U	Attiva il flag della routine utente
?	Richiama il disassemblatore (TV.DIS o PR.DIS)

Avendo assegnato le funzioni a questi tasti, sarà possibile attivare o disattivare la stampante digitando P e, similmente per la routine utente, digitando U. Potremo visualizzare o stampare un dump di memoria digitando H e stampare o visualizzare un listato disassemblato digitando ?. Potremo ancora trasferire blocchi di programma da una locazione di memoria ad un'altra digitando M e richiamare il text editor digitando T.
Di seguito è riportato il listato della routine che permetterà di ottenere queste prestazioni e, siccome vogliamo espandere il Monitor Visibile, la chiameremo ESPAND.

ESPAND

```
                              Quando ESPAND viene richiamata,
                              il carattere digitato si trova nell'ac-
                              cumulatore
ESPAND    CMP #'P             È "P"?
          BNE IF.U            Se no, esegue il test successivo
          LDA PRINTR          Se sì, attiva la stampante
          EOR #$FF
          STA PRINTR
          RTS
IF.U      CMP #'U             È "U"?
          BNE IF.H            Se no, esegue il test successivo
          LDA USR.FN          Se sì, attiva la routine utente
          EOR #$FF
          STA USR.FN
          RTS
IF.H      CMP #'H             È "H"?
          BNE IF.M            Se no, esegue il test successivo
          LDA PRINTR          È attiva la stampante?
          BNE NEXT.1          Se sì, richiama PR.DUMP
          JSR TVDUMP          Se no, richiama TVDUMP
          RTS
NEXT.1    JSR PRDUMP
          RTS
IF.M      CMP #'M             È "M"?
          BNE IF.DIS          Se no, esegue il test successivo
          JSR MOVER           Se sì, richiama l'utility di trasferi-
          RTS                 mento
IF.DIS    CMP #'?             È "?"
          BNE IF.T            Se no, esegue il test successivo
          LDA PRINTR          È attiva la stampante?
          BNE NEXT.2          Se sì, stampa il listato disassemblato
          JSR TV.DIS          Se no, visualizza su video
          RTS
NEXT.2    JSR PR.DIS
          RTS
IF.T      CMP #'T             È "T"?
          BNE FINE            Se no ... ritorna
          JSR EDITOR          Se sì, richiama l'editor
          RTS
FINE      RTS                 È possibile espandere ulteriormen-
                              te il Monitor Visibile aggiungendo
                              altri test in questo punto
```

L'unica cosa che resta da fare è modificare la routine AGGIOR in modo che richiami ESPAND piuttosto che COMOD.
Attualmente la routine AGGIOR del Monitor Visibile, prima di uscire richiama COMOD con i byte $20, $10, $30 agli indirizzi $33D1, $33D2 e $33D3 rispettivamente. Per fare in modo che sia richiamata ESTEND basterà cambiare la locazione $33D2 da $10 a $B0.
Possiamo cambiare questa locazione usando il monitor non esteso e facendo attenzione a usare solamente i tasti assegnati. Una volta cambiato il valore della locazione $33D2 si potrà digitare qualunque carattere senza più problemi e quindi provare le nuove funzioni incluse nel Monitor Visibile integrato premendo i tasti, ora attivi. H, M, P, U, ? e T.

Capitolo 13
Caricamento del software

Dal Capitolo 5 al Capitolo 12 abbiamo proposto del software che potrà esservi certamente utile solo se sarete in grado di immetterlo nel vostro computer. Se il vostro C-64 o VIC-20 avesse un monitor sarebbe possibile inserire i programmi senza alcuna difficoltà (ma se ci fosse un monitor residente non si sarebbe sentita la necessità di questo libro!).
Il vostro computer possiede un interprete BASIC in ROM ma non un monitor di codice macchina. Come sarà possibile immettere il codice oggetto in memoria servendosi esclusivamente dell'interprete BASIC? E, ammesso di compiere l'operazione con successo, come sarà possibile registrare su disco o cassetta il programma in codice macchina? Se possediamo un interprete BASIC, la soluzione più semplice è che il nostro codice oggetto assomigli il più possibile a un programma BASIC.
Questo non è difficile; un programma BASIC può usare l'istruzione DATA, quindi potremo studiare un programma in BASIC contenente un certo numero di istruzioni DATA che rappresentino, in decimale, il valore di successioni di byte del codice oggetto. Poi, il programma BASIC leggerà i DATA per mezzo dell'istruzione READ e inserirà questi valori nell'appropriata area di memoria tramite l'istruzione POKE.

Uso del BASIC per caricare il codice macchina

Il software contenuto in questo libro può essere immesso in memoria semplicemente tramite una serie di programmi BASIC. Ogni programma BASIC sarà costituito da un LOADER del codice oggetto e da un certo

numero di istruzioni DATA. Le prime due specificano l'intervallo delle istruzioni DATA che seguono. Ogni DATA che segue è composto da dieci valori; il primo rappresenta l'indirizzo dove caricare il codice oggetto della linea, gli otto valori successivi rappresentano il codice oggetto e il decimo è la somma di controllo (*checksum*). La somma di controllo non è altro che il totale dei primi nove valori. Di questi dieci valori quindi il primo e l'ultimo saranno sempre maggiori di 4000 mentre gli altri saranno sempre minori di 256.
Le Appendici da E1 a E11 contengono il programma del libro sotto questa forma. Dovrete digitare ognuno di questi DATA e il programma vi segnalerà eventuali errori di battitura per mezzo della somma di controllo. Infatti, commettendo errori di battitura nelle linee DATA, difficilmente la somma di controllo corrisponderà alla somma dei valori inseriti e quando il LOADER verrà eseguito segnalerà il numero della linea dove si è verificato l'errore.
Il LOADER farà uso delle seguenti variabili:

A	L'indirizzo specificato dal primo valore della linea DATA
BYTE	Un array che contiene i valori degli otto byte consecutivi di codice oggetto specificati dalla linea DATA
CHECK	La somma di controllo specificata dalla linea DATA
PRIMA	Il numero di linea della prima istruzione DATA contenente il codice oggetto
ULTIMA	Il numero di linea dell'ultima istruzione DATA contenente il codice oggetto
LINE	Un contatore di linea, che controlla il numero delle linee di dati del codice oggetto già caricate in memoria
SUM	La somma degli 8 valori e dell'indirizzo specificato da una linea DATA. Se questa somma è uguale a quella di controllo probabilmente non ci sono errori di battitura
TEMP	Una variabile di comodo

NOTA: nel listato che segue è possibile omettere le REM.

```
100 REM                      LOADER di Ken Skier
110 REM
120 DIM BYTE (8)             :REM inizializza l'array BYTE
130 READ PRIMA               :REM legge il numero della
                             prima linea
140 REM                      che contiene il codice oggetto
150 READ ULTIMA              :REM legge il numero dell'ul-
                             tima linea
160 REM                      che contiene il codice oggetto
```

```
170 FOR LINE=PRIMA TO ULTIMA   :REM legge le linee DATA spe-
                               cificate
180 GOSUB 300                  :REM carica in memoria i DA-
190 NEXT LINE                  TA specificati
200 PRINT "HO CARICATO SENZA ERRORI DALLA LINEA ";
    PRIMA; " ALLA LINEA ";ULTIMA
210 END                        :REM se alla fine è tutto ok lo
                               segnala
220 REM
230 REM
300 READ A                     :REM legge l'indirizzo
310 SUM=A                      :REM inizia il calcolo della
                               somma
320 FOR J=1 TO 8               :REM legge 8 byte dai DATA
330 READ BYTE(J)
340 SUM=SUM+BYTE(J)            :REM continua il calcolo della
                               somma
350 NEXT J
370 READ CHECK                 :REM legge la checksum dalla
                               linea
380 IF SUM < > CHECK THEN 500  :REM se differente da quella
381 REM                        calcolata lo segnala
390 FOR J=1 TO 8               :REM se non ci sono errori
391 REM                        memorizza il codice oggetto
400 POKE A+J-1, BYTE(J)
410 NEXT J
420 RETURN                     :REM torna a leggere gli otto
421 REM                        valori contenuti nella linea
                               successiva
490 REM                        segnalazione errori
500 PRINT "CHECKSUM ERRATA ALLA LINEA ";LINE
510 PRINT "INDIRIZZO SPECIFICATO= ";A
520 END
530 REM
540 REM
550 DATA xxx                   :REM xxx è il numero della
560 REM                        prima linea di DATA che con-
570 REM                        tiene il codice oggetto
580 REM
590 DATA yyy                   :REM yyy è il numero dell'ul-
600 REM                        tima linea di DATA che con-
620 REM                        tiene il codice oggetto
```

Dopo aver digitato il LOADER sarà utile salvarlo su cassetta o disco.
A questo punto potremo aggiungere i DATA copiandoli da una delle Appendici E1-E11. È assolutamente sconsigliabile aggiungere linee DATA copiate da più di un'appendice. Bisognerà invece fare attenzione a non superare con il LOADER e i DATA la locazione di memoria $2FFF (12287 decimale). Per essere sicuri di non superare questo limite sarà sufficiente spostare il limite di memoria disponibile a un programma BASIC. Si potrà ottenere ciò per mezzo delle istruzioni BASIC seguenti che dovranno essere digitate immediatamente dopo l'accensione del computer:

```
POKE 52,47 : POKE 56,47 : POKE 51,255 : POKE 55,255
```

Aggiungendo le linee DATA al LOADER si verificherà la situazione in cui il computer avverte di non avere più memoria disponibile (OUT OF MEMORY). Se ciò avviene, cancellate l'ultima linea DATA inserita. Supponiamo, per esempio, di aver inserito le linee DATA da 1000 a 1022 quando avviene la segnalazione. Cancellate la linea 1022 e inserite i numeri della prima e ultima linea inserita nei DATA che si trovano alle linee 550 e 590

```
550 DATA 1000
590 DATA 1021
```

Queste due linee permetteranno al LOADER di sapere quante linee DATA di codice oggetto leggere (nel nostro esempio 22 e precisamente della 1000 alla 1021). Facendo partire il LOADER se ci sono errori di battitura verrà visualizzato il messaggio di errore

```
CHECKSUM ERRATA ALLA LINEA 1021
INDIRIZZO SPECIFICATO 12640
```

Questo significa che alla linea 1021 i dati digitati non sono corretti e, per facilitarvi ancora di più nella ricerca della linea errata, la seconda riga del messaggio specifica che il primo dato della linea errata è 12640. Avendo a disposizione questi due dati sarà sufficiente confrontare la linea specificata con il listato dell'appendice e correggere i valori che non corrispondono.
Se invece tutte le linee DATA sono state digitate correttamente, il programma visualizzerà il messaggio

```
HO CARICATO SENZA ERRORI DALLA LINEA xxx ALLA LINEA yyy
```

dove xxx rappresenta il numero della prima linea DATA che contiene il codice oggetto e yyy il numero dell'ultima. Quando otterrete questo messaggio, potrete salvare il LOADER su cassetta o disco, richiamare un

LOADER "pulito" e proseguire con le successive linee di DATA. Dopo aver digitato tutti i DATI delle Appendici E1-E11 e quelli delle Appendici E12 (per il VIC-20) o E13 (per il C-64), eseguendo tutti i programmi LOADER salvati man mano, avrete disponibile in memoria il Monitor Visibile.

Attivazione del Monitor Visibile

Dopo aver caricato tutte le routine che compongono il Monitor Visibile e i dati di sistema, sarà possibile attivare il Monitor Visibile facendo eseguire al 6502 un salto alla locazione $308F che corrisponde in decimale a 12431. Per eseguire questo salto sarà sufficiente digitare il seguente comando BASIC:

```
SYS (12431)
```

Se tutte le operazioni sono state eseguite correttamente, dovrebbe apparire la videata del Monitor Visibile. Potrete a questo punto provare a premere la barra spaziatrice per controllare che i valori visualizzati nei campi 1 e 2 cambino oppure provare a sperimentare tutte le altre funzioni del Monitor Visibile.
Se si verificasse qualche anomalia di funzionamento, ricontrollate tutte le linee DATA aggiunte ai vari LOADER perché anche un solo errore nel compiere questa operazione porterebbe sicuramente a problemi di questo tipo del Monitor Visibile.
Per estendere il Monitor Visibile come descritto nel Capitolo 12 basterà memorizzare $B0 alla locazione $33D2, mentre per disabilitare le estensioni sarà sufficiente inserire il valore $10. A questo punto sarete in grado di esplorare da vicino il contenuto della memoria del sistema e di modificarlo. Con questa possibilità sarete veramente padroni del sistema.

Salvataggio di un programma in codice macchina

Con l'aiuto del Monitor Visibile potrete creare programmi in codice macchina veramente potenti. Vorrete però a questo punto poter salvare il vostro lavoro su nastro o disco, in modo da poterlo richiamare all'occorrenza. Fortunatamente il C-64 e il VIC-20 posseggono una serie di routine in ROM, chiamate KERNAL, che permettono di salvare il codice oggetto su nastro o disco. Supponiamo che il Monitor Visibile integrato sia in memoria come pure il programma oggetto che avete sviluppato. Sarà possi-

bile salvare il vostro programma in codice oggetto per mezzo del seguente programma BASIC:

Programma BASIC per salvare codice macchina su nastro o disco

```
10  UNITA=12364
20  LUNG=12365
30  NOME=12366
40  MLSAV=12386
50  SETADS=13795
60  :
100 PRINT "SALVATAGGIO CODICE MACCHINA"
110 PRINT
120 INPUT "NOME DEL FILE";NAME$
125 IF LEN(NAME$)>19 THEN NAME$=LEFT$(NAME$,19)
130 POKE LUNG,LEN(NAME$)
140 IF LEN(NAME$)=0 THEN 200
150 :
160 FOR J=1 TO LEN(NAME$)
170 :  POKE NAME+J-1,ASC(MID$(NAME$,J))
180 NEXT J
190 :
200 PRINT "NASTRO O DISCO? (N/D)";
210 GET A$:IF LEN (A$)=0 THEN 210
220 IF A$="N" THEN POKE UNITA,1:GOTO 300
230 IF A$="D" THEN POKE UNITA,8:GOTO 300
240 PRINT "TASTO ERRATO":PRINT:GOTO 200
250 :
300 SYS (SETADS) : REM SETTA GLI INDIRIZZI DI PARTENZA E
    DI FINE
310 SYS (MLSAV) : REM SALVA IL PROGRAMMA SU DISCO O
    NASTRO
```

Questo programma BASIC chiederà di specificare il nome del file che volete salvare e a che periferica accedere (nastro o disco).
Successivamente per poter caricare un programma salvato in questa maniera sarà sufficiente digitare:

LOAD"nome del file",1,1

se il programma è su nastro, mentre se si trova su disco:

LOAD"nome del file" ",8,1

Dopo aver caricato un programma in codice macchina potrebbero essersi modificati alcuni puntatori utilizzati dal BASIC e potreste ottenere messaggi di errore per comandi perfettamente legittimi. Per eliminare questo problema basterà digitare, dopo aver caricato il codice oggetto, i seguenti comandi:

```
POKE 52,47 : POKE 56,47 : POKE 51,255 : POKE 55,255
NEW
```

Con questi semplici comandi verrà cancellato il programma BASIC che potrebbe essere in memoria, ma non il codice oggetto che risiede in memoria nelle locazioni superiori a $2FFF.

Appendice

Codice ASCII e set di istruzioni A

A.1 Codice ASCII dei caratteri

Codice	Carattere	Codice	Carattere	Codice	Carattere	Codice	Carattere
00	NUL	10	DLE	20	SP	30	0
01	SOH	11	DC1	21	!	31	1
02	STX	12	DC2	22	"	32	2
03	ETX	13	DC3	23	#	33	3
04	EOT	14	DC4	24	$	34	4
05	ENQ	15	NAK	25	%	35	5
06	ACK	16	SYN	26	&	36	6
07	BEL	17	ETB	27	'	37	7
08	BS	18	CAN	28	(	38	8
09	HT	19	EM	29	)	39	9
0A	LF	1A	SUB	2A	*	3A	:
0B	VT	1B	ESC	2B	+	3B	;
0C	FF	1C	FS	2C	,	3C	<
0D	CR	1D	GS	2D	−	3D	=
0E	SO	1E	RS	2E	.	3E	>
0F	SI	1F	US	2F	/	3F	?

40	@	50	P	60	`	70	p
41	A	51	Q	61	a	71	q
42	B	52	R	62	b	72	r
43	C	53	S	63	c	73	s
44	D	54	T	64	d	74	t
45	E	55	U	65	e	75	u
46	F	56	V	66	f	76	v
47	G	57	W	67	g	77	w
48	H	58	X	68	h	78	x
49	I	59	Y	69	i	79	y
4A	J	5A	Z	6A	j	7A	z
4B	K	5B	[	6B	k	7B	{
4C	L	5C	\	6C	l	7C	¦
4D	M	5D	]	6D	m	7D	}
4E	N	5E	^	6E	n	7E	~
4F	O	5F	_	6F	o	7F	DEL

A.2 Istruzioni del 6502
Lista degli mnemonici

ADC	Somma il contenuto della memoria all'accumulatore con carry
AND	AND tra memoria e accumulatore
ASL	Shifta un bit a sinistra (memoria o accumulatore)
BCC	Salta se il carry è zero
BCS	Salta se il carry è uno
BEQ	Salta se il risultato è zero
BIT	Testa i bit in memoria con l'accumulatore
BMI	Salta se il risultato è negativo
BNE	Salta se il risultato è diverso da zero
BPL	Salta se il risultato è positivo
BRK	Break forzato
BVC	Salta se l'overflow è zero
BVS	Salta se l'overflow è uno
CLC	Azzera il flag Carry
CLD	Azzera il modo decimale
CLI	Azzera il bit di interrupt
CLV	Azzera il flag Overflow
CMP	Compara la memoria con l'accumulatore
CPX	Compara la memoria con il registro X
CPY	Compara la memoria con il registro Y
DEC	Decrementa in memoria
DEX	Decrementa il registro X
DEY	Decrementa il registro Y
EOR	OR esclusivo tra memoria e accumulatore
INC	Incrementa in memoria
INX	Incrementa il registro X
INY	Incrementa il registro Y
JMP	Salta alla locazione
JSR	Salta salvando l'indirizzo di ritorno
LDA	Carica l'accumulatore con la memoria
LDX	Carica il registro X con la memoria
LDY	Carica il registro Y con la memoria
LSR	Shifta un bit a destra (memoria o accumulatore)

NOP	Nessuna operazione
ORA	OR tra memoria e accumulatore
PHA	Salva l'accumulatore nello stack
PHP	Salva il registro P nello stack
PLA	Legge l'accumulatore dallo stack
PLP	Legge il registro P dallo stack
ROL	Ruota di un bit a sinistra (memoria o accumulatore)
ROR	Ruota di un bit a destra (memoria o accumulatore)
RTI	Ritorna dall'interrupt
RTS	Ritorna da subroutine
SBC	Sottrae la memoria dall'accumulatore con riporto
SEC	Setta il flag Carry
SED	Seleziona il modo decimale
SEI	Disabilita l'interrupt
STA	Memorizza l'accumulatore in memoria
STX	Memorizza il registro X in memoria
STY	Memorizza il registro Y in memoria
TAX	Trasferisce l'accumulatore al registro X
TAY	Trasferisce l'accumulatore al registro Y
TSX	Trasferisce il puntatore dello stack in X
TXA	Trasferisce il registro X nell'accumulatore
TXS	Trasferisce il registro X nel puntatore dello stack
TYA	Trasferisce il registro Y nell'accumulatore

A.3 Istruzioni del 6502 Lista degli opcode

00 — BRK
01 — ORA — (Indiretto,X)
02 — *
03 — *
04 — *
05 — ORA — Pagina zero
06 — ASL — Pagina zero
07 — *
08 — PHP
09 — ORA — Immediato
0A — ASL — Accumulatore
0B — *
0C — *
0D — ORA — Assoluto
0E — ASL — Assoluto
0F — *

10 — BPL
11 — ORA — (Indiretto),Y
12 — *
13 — *
14 — *
15 — ORA — Pagina zero,X
16 — ASL — Pagina zero,X
17 — *
18 — CLC
19 — ORA — Assoluto,Y
1A — *
1B — *
1C — *
1D — ORA — Assoluto,X
1E — *
1F — *

20 — JSR
21 — AND — (Indiretto,X)
22 — *
23 — *
24 — Bit — Pagina zero
25 — AND — Pagina zero
26 — ROL — Pagina zero
27 — *
28 — PLP
29 — AND — Immediato
2A — ROL — Accumulatore
2B — *
2C — BIT — Assoluto
2D — AND — Assoluto
2E — ROL — Assoluto
2F — *

30 — BMI
31 — AND — (Indiretto),Y
32 — *
33 — *
34 — *
35 — AND — Pagina zero,X
36 — ROL — Pagina zero,X
37 *
38 — SEC
39 — AND — Assoluto,Y
3A — *
3B — *
3C — *
3D — AND — Assoluto,X
3F — *

40 — RTI
41 — EOR — (Indiretto,X)
42 — *
43 — *
44 — *
45 — EOR — Pagina zero
46 — LSR — Pagina zero

*Opcode disponibili per futuri sviluppi

47 — *
48 — PHA
49 — EOR — Immediato
4A — LSR — Accumulatore
4B — *
4C — JMP — Assoluto
4D — EOR — Assoluto
4E — LSR — Assoluto
4F — *

50 — BVC
51 — EOR — (Indiretto),Y
52 — *
53 — *
54 — *
55 — EOR — Pagina zero,X
56 — Pagina zero,X
57 — *
58 — CLI
59 — EOR — Assoluto,Y
5A — *
5B — *
5C — *
5D — EOR — Assoluto,X
5E — LSR — Assoluto,X
5F — *

60 — RTS
61 — ADC — (Indiretto,X)
62 — *
63 — *
64 — *
65 — ADC — Pagina zero
66 — ROR — Pagina zero
67 — *
68 — PLA
69 — ADC — Immediato
6A — ROR — Accumulatore
6B — *
6C — JMP — Indiretto
6D — ADC — Assoluto
6E — ROR — Assoluto
6F — *

70 — BVS
71 — ADC — (Indiretto),Y
72 — *
73 — *
74 — *
75 — ADC — Pagina zero,X
76 — ROR — Pagina zero,X
77 — *
78 — SEI
79 — ADC — Assoluto,Y
7A — *
7B — *
7C — *
7D — ADC — Assoluto,X
7E — ROR — Assoluto,X
7F — *

80 — *
81 — STA — (Indiretto,X)
82 — *
83 — *
84 — STY — Pagina zero
85 — STA — Pagina zero
86 — STX — Pagina zero
87 — *
88 — DEY
89 — *
8A — TXA
8B — *
8C — STY — Assoluto
8D — STA — Assoluto
8E — STX — Assoluto
8F — *

90 — BCC

*Opcode disponibili per futuri sviluppi

91 — STA — (Indiretto),Y
92 — *
93 — *
94 — STY — Pagina zero,X
95 — STA — Pagina zero,X
96 — STX — Pagina zero,Y
97 — *
98 — TYA
99 — STA — Assoluto,Y
9A — TXS
9B — *
9C — *
9D — STA — Assoluto,X
9E — *
9F — *

A0 — LDY — Immediato
A1 — LDA — (Indiretto,X)
A2 — LDX — Immediato
A3 — *
A4 — LDY — Pagina zero
A5 — LDA — Pagina zero
A6 — LDX — Pagina zero
A7 — *
A8 — TAY
A9 — LDA — Immediato
AA — TAX
AB — *
AC — LDY — Assoluto
AD — LDA — Assoluto
AE — LDX — Assoluto
AF — *

B0 — BCS
B1 — LDA — (Indiretto),Y
B2 — *
B3 — *
B4 — LDY — Pagina zero,X
B5 — LDA — Pagina zero,X
B6 — LDX — Pagina zero,Y
B7 — *
B8 — CLV
B9 — LDA — Assoluto,Y
BA — TSX
BB — *
BC — LDY — Assoluto,X
BD — LDA — Assoluto,X
BE — LDX — Assoluto,Y
BF — *

C0 — CPY — Immediato
C1 — CMP — (Indiretto,X)
C2 — *
C3 — *
C4 — CPY — Pagina zero
C5 — CMP — Pagina zero
C6 — DEC — Pagina zero
C7 — *
C8 — INY
C9 — CMP — Immediato
CA — DEX
CB — *
CC — CPY — Assoluto
CD — CMP — Assoluto
CE — DEC — Assoluto
CF — *

D0 — BNE
D1 — CMP — (Indiretto),Y
D2 — *
D3 — *
D4 — *
D5 — CMP — Pagina zero,X
D6 — DEC — Pagina zero,X
D7 — *
D8 — CLD
D9 — CMP — Assoluto,Y
DA — *
DB — *
DC — *

*Opcode disponibili per futuri sviluppi

DD — CMP — Assoluto,X
DE — DEC — Assoluto,X
DF — *

E0 — CPX — Immediato
E1 — SEC — (Indiretto,X)
E2 — *
E3 — *
E4 — CPX — Pagina zero
E5 — SBC — Pagina zero
E6 — INC — Pagina zero
E7 — *
E8 — INX
E9 — SBC — Immediato
EA — NOP
EB — *
EC — CPX — Assoluto
ED — SBC — Assoluto
EE — INC — Assoluto
EF — *

F0 — BEQ
F1 — SBC — (Indiretto),Y
F2 — *
F3 — *
F4 — *
F5 — SBC — Pagina zero,X
F6 — INC — Pagina zero,X
F7 — *
F8 — SED
F9 — SBC — Assoluto,Y
FA — *
FB — *
FC — *
FD — SBC — Assoluto,X
FE — INC — Assoluto,X
FF — *

*Opcode disponibili per futuri sviluppi

A.4 Tempo di esecuzione delle istruzioni

	Accumulatore	Immediato	Pagina zero	Pagina zero,X	Pagina zero,Y	Assoluto	Assoluto,X	Assoluto,Y	Implicito	Relativo	(Indiretto),X	(Indiretto),Y	Indiretto assoluto
ADC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
AND	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
ASL	2	.	5	6	.	6	7	.	.	.	.	.	.
BCC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BCS	.	.	.	.	.	.	.	.	.	2**	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	2**	.	.	.
BIT	.	.	3	.	.	4	.	.	.	.	.	.	.
BMI	.	.	.	.	.	.	.	.	.	2**	.	.	.
BNE	.	.	.	.	.	.	.	.	.	2**	.	.	.
BPL	.	.	.	.	.	.	.	.	.	2**	.	.	.
BRK	.	.	.	.	.	.	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BVS	.	.	.	.	.	.	.	.	.	2**	.	.	.
CLC	.	.	.	.	.	.	.	.	2	.	.	.	.
CLD	.	.	.	.	.	.	.	.	2	.	.	.	.
CLI	.	.	.	.	.	.	.	.	2	.	.	.	.
CLV	.	.	.	.	.	.	.	.	2	.	.	.	.
CMP	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
CPX	.	2	3	.	.	4	.	.	.	.	.	.	.
CPY	.	2	3	.	.	4	.	.	.	.	.	.	.
DEC	.	.	5	6	.	6	7	.	.	.	.	.	.
DEX	.	.	.	.	.	.	.	.	2	.	.	.	.
DEY	.	.	.	.	.	.	.	.	2	.	.	.	.
EOR	.	2	3	4	.	4	4*	4*	.	.	6	5	.

Le cifre indicano il numero di cicli macchina necessari all'esecuzione delle singole istruzioni.

	Accumulatore	Immediato	Pagina zero	Pagina zero,X	Pagina zero,Y	Assoluto	Assoluto,X	Assoluto,Y	Implicito	Relativo	(Indiretto),X	(Indiretto),Y	Indiretto assoluto
INC	.	.	5	6	.	6	7	.	.	.	.	.	.
INX	.	.	.	.	.	.	.	.	2	.	.	.	.
INY	.	.	.	.	.	.	.	.	2	.	.	.	.
JMP	.	.	.	.	.	3	.	.	.	.	.	.	5
JSR	.	.	.	.	.	6	.	.	.	.	.	.	.
LDA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
LDX	.	2	3	.	4	4	.	4*	.	.	.	.	.
LDY	.	2	3	4	.	4	4*	.	.	.	.	.	.
LSR	2	.	5	6	.	6	7	.	.	.	.	.	.
NOP	.	.	.	.	.	.	.	.	2	.	.	.	.
ORA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
PHA	.	.	.	.	.	.	.	.	3	.	.	.	.
PHP	.	.	.	.	.	.	.	.	3	.	.	.	.
PLA	.	.	.	.	.	.	.	.	4	.	.	.	.
PLP	.	.	.	.	.	.	.	.	4	.	.	.	.
ROL	2	.	5	6	.	6	7	.	.	.	.	.	.
ROR	2	.	5	6	.	6	7	.	.	.	.	.	.
RTI	.	.	.	.	.	.	.	.	6	.	.	.	.
RTS	.	.	.	.	.	.	.	.	6	.	.	.	.
SBC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
SEC	.	.	.	.	.	.	.	.	2	.	.	.	.
SED	.	.	.	.	.	.	.	.	2	.	.	.	.
SEI	.	.	.	.	.	.	.	.	.	.	.	.	.
STA	.	.	3	4	.	4	5	5	.	.	6	6	.
STX*	.	.	3	.	4	4	.	.	.	.	.	.	.
STY**	.	.	3	4	.	4	.	.	.	.	.	.	.
TAX	.	.	.	.	.	.	.	.	2	.	.	.	.
TAY	.	.	.	.	.	.	.	.	2	.	.	.	.
TSX	.	.	.	.	.	.	.	.	2	.	.	.	.
TXA	.	.	.	.	.	.	.	.	2	.	.	.	.
TXS	.	.	.	.	.	.	.	.	2	.	.	.	.
TYA	.	.	.	.	.	.	.	.	2	.	.	.	.

*Aggiungere un ciclo macchina se l'indice di indirizzamento riguarda 2 pagine di memoria differenti

**Aggiungere un ciclo se il salto viene effettuato. Aggiungere un ulteriore ciclo se il salto riguarda pagine diverse

A.5 Opcode del 6502 in ordine alfabetico e di indirizzamento

Modi di indirizzamento

	Assoluto	Assoluto,X	Assoluto,Y	Accumulatore	Immediato	Implicito	Indiretto	Indiretto,X	Indiretto,Y	Relativo	Pagina zero	Pagina zero,X	Pagina zero,Y
ADC	6D	7D	79	.	69	.	.	61	71	.	65	75	.
AND	2D	3D	39	.	29	.	.	21	31	.	25	35	.
ASL	0E	1E	.	0A	.	.	.	.	.	.	06	16	.
BCC	.	.	.	.	.	.	.	.	.	90	.	.	.
BCS	.	.	.	.	.	.	.	.	.	B0	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	F0	.	.	.
BIT	2C	.	.	.	.	.	.	.	.	.	24	.	.
BMI	.	.	.	.	.	.	.	.	.	30	.	.	.
BNE	.	.	.	.	.	.	.	.	.	D0	.	.	.
BPL	.	.	.	.	.	.	.	.	.	10	.	.	.
BRK	.	.	.	.	.	00	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	50	.	.	.
BVS	.	.	.	.	.	.	.	.	.	70	.	.	.
CLC	.	.	.	.	.	18	.	.	.	.	.	.	.
CLD	.	.	.	.	.	D8	.	.	.	.	.	.	.
CLI	.	.	.	.	.	58	.	.	.	.	.	.	.

	Assoluto	Assoluto,X	Assoluto,Y	Accumulatore	Immediato	Implicito	Indiretto	Indiretto,X	Indiretto,Y	Relativo	Pagina zero	Pagina zero,X	Pagina zero,Y
CLV	.	.	.	.	.	B8	.	.	.	.	.	.	.
CMP	CD	DD	D9	.	C9	.	.	C1	D1	.	C5	D5	.
CPX	EC	.	.	.	E0	.	.	.	.	.	E4	.	.
CPY	CC	.	.	.	C0	.	.	.	.	.	C4	.	.
DEC	CE	DE	.	.	.	.	.	.	.	.	C6	D6	.
DEX	.	.	.	.	.	CA	.	.	.	.	.	.	.
DEY	.	.	.	.	.	88	.	.	.	.	.	.	.
EOR	4D	5D	59	.	49	.	.	41	51	.	45	55	.
INC	EE	FE	.	.	.	.	.	.	.	.	E6	F6	.
INX	.	.	.	.	.	E8	.	.	.	.	.	.	.
INY	.	.	.	.	.	C8	.	.	.	.	.	.	.
JMP	4C	.	.	.	.	.	6C	.	.	.	.	.	.
JSR	20	.	.	.	.	.	.	.	.	.	.	.	.
LDA	AD	BD	B9	.	A9	.	.	A1	B1	.	A5	B5	.
LDX	AE	.	BE	.	A2	.	.	.	.	.	A6	.	.
LDY	AC	BC	.	.	A0	.	.	.	.	.	A4	B4	.
LSR	4E	5E	.	4A	.	.	.	.	.	.	46	56	.
NOP	.	.	.	.	.	EA	.	.	.	.	.	.	.
ORA	0D	1D	19	.	09	.	.	01	11	.	05	15	.
PHA	.	.	.	.	.	48	.	.	.	.	.	.	.
PHP	.	.	.	.	.	08	.	.	.	.	.	.	.
PLA	.	.	.	.	.	68	.	.	.	.	.	.	.
PLP	.	.	.	.	.	28	.	.	.	.	.	.	.
ROL	2E	3E	.	2A	.	.	.	.	.	.	26	36	.
ROR	6E	7E	.	6A	.	.	.	.	.	.	66	76	.
RTI	.	.	.	.	.	40	.	.	.	.	.	.	.

	Assoluto	Assoluto,X	Assoluto,Y	Accumulatore	Immediato	Implicito	Indiretto	Indiretto,X	Indiretto,Y	Relativo	Pagina zero	Pagina zero,X	Pagina zero,Y
RTS	.	.	.	.	.	60	.	.	.	.	.	.	.
SBC	ED	FD	F9	.	E9	.	.	E1	F1	.	E5	F5	.
SEC	.	.	.	.	.	38	.	.	.	.	.	.	.
SED	.	.	.	.	.	F8	.	.	.	.	.	.	.
SEI	.	.	.	.	.	78	.	.	.	.	.	.	.
STA	8D	9D	99	.	.	.	.	81	91	.	85	95	.
STX	8E	.	.	.	.	.	.	.	.	.	86	.	.
STY	8C	.	.	.	.	.	.	.	.	.	84	94	.
TAX	.	.	.	.	.	AA	.	.	.	.	.	.	.
TAY	.	.	.	.	.	A8	.	.	.	.	.	.	.
TSX	.	.	.	.	.	BA	.	.	.	.	.	.	.
TXA	.	.	.	.	.	8A	.	.	.	.	.	.	.
TXS	.	.	.	.	.	9A	.	.	.	.	.	.	.
TYA	.	.	.	.	.	98	.	.	.	.	.	.	.

Appendice **B**

Istruzioni per l'implementazione

B.1 Il VIC-20

Il VIC-20 è un computer piccolo ed economico. Sfortunatamente però la versione più economica non possiede sufficiente memoria per il software presentato in questo libro. Il VIC-20 senza espansione possiede solamente 3K di RAM mentre per poter usare i programmi descritti occorrono almeno 8K di espansione RAM a partire dall'indirizzo $2000. È possibile aggiungere estensione di memoria installando nel VIC l'espansione Commodore VIC-1110 da 8K oppure la VIC-1111 da 16K.
Se il vostro VIC arriva ad avere 8K di RAM a partire da $2000, la memoria video è mappata a $1000 (nel VIC senza espansione è a $1E00). Lo schermo contiene 25 righe di 22 caratteri e l'indirizzo di ogni locazione video è 22 ($16) volte maggiore della locazione precedente. Quindi i parametri-schermo per il VIC-20 sono:

HOME	.WORD $1000	
RIGINC	.BYTE 22	Differenza di indirizzo tra una riga e la successiva
TVCOL	.BYTE 21	
TVRIG	.BYTE 24	

Questo però non è tutto ciò che ci occorre sapere circa lo schermo del VIC. In alcuni computer è possibile visualizzare un carattere in una determinata posizione semplicemente memorizzando il codice ASCII del carattere alla locazione della posizione voluta. Visualizzare caratteri ASCII

sul VIC è leggermente più complicato. Prima di tutto c'è da considerare la memoria del colore, poi c'è il problema di determinare l'appropriato codice schermo del VIC per un dato carattere.

Memoria colore

Oltre alla memoria video il VIC riserva anche una zona di memoria chiamata *memoria colore*. Ogni byte nella memoria video ha un corrispondente nella memoria colore. La memoria video specifica il carattere da visualizzare e, naturalmente, la memoria colore specifica il colore con cui visualizzare quel carattere. Nel VIC con espansione, la memoria colore si trova a partire dalla locazione $9400 cioè $8400 byte sopra le memoria video, quindi l'n-esimo byte della memoria colore definisce il colore dell'n-esimo carattere della memoria video.
Notate che se una parte della memoria colore contiene il codice del colore di sfondo, nessun carattere sarà visibile nella corrispondente porzione dello schermo, anche se esiste del testo in questa porzione. Quindi, prima di visualizzare un carattere sarà necessario memorizzare l'appropriato codice colore nella memoria colore.
Che colore dovremo usare? Potremo usare un colore arbitrario, per esempio il nero, ma sarebbe più sensato usare il colore selezionato dall'utente. La locazione $286 (646 decimale) contiene il codice colore selezionato dall'utente, quindi prima di visualizzare un carattere leggeremo il codice colore nella locazione $286 e memorizzeremo questo codice nell'appropriata locazione della memoria colore. Questa locazione sarà esattamente $8400 byte più in alto della locazione a cui punta TV.PTR.
Sappiamo ora come predisporre la memoria colore quando desideriamo visualizzare del testo nello schermo, ma come determinare quali sono gli appropriati codici schermo da usare?

Codici schermo

Per visualizzare un dato carattere sullo schermo bisogna memorizzare il codice appropriato in memoria. La Tabella B1.1 mostra i *codici schermo* del VIC-20. In questa tabella tutti i caratteri grafici sono rappresentati da un trattino, ma se volete visualizzarli sarà sufficiente eseguire il programma seguente.

```
100 REM VISUALIZZA I CODICI SCHERMO DEL VIC-20
110 REM IN UNA MATRICE 16×16
120 REM
```

```
130 PRINT CHR$(147) : REM PULISCE LO SCHERMO
140 VIDEO=4096  : REM MEMORIA VIDEO
150 CMEM=37888 : REM MEMORIA COLORE
160 C=PEEK(646)  : REM COLORE SELEZIONATO
170 REM
180 FOR RIG=0 TO 15
190 FOR COL=0 TO 15
200 POKE CMEM+COL+22*RIG,C
210 POKE VIDEO+COL+22*RIG,COL+16*RIG
220 NEXT COL,RIG
230 REM
240 GOTO 240
```

Questo programma pulisce lo schermo, memorizza il codice colore selezionato nella memoria colore e inserisce tutti i 256 codici schermo in una griglia 16 per 16 della memoria video. Fatto questo continuerà in un loop senza fine alla linea 240. Per uscirne sarà sufficiente premere il tasto RUN/STOP.

Tabella B1.1 Set di caratteri del VIC-20

	Byte basso															
Byte alto	**-0**	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**	**-7**	**-8**	**-9**	**-A**	**-B**	**-C**	**-D**	**-E**	**-F**
0-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
2-		!	"	#	$	%	&	'	(	)	*	+	'	–	.	/
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4-	–	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
5-	p	q	r	s	t	u	v	w	x	y	z	—	—	—	—	—
6-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
7-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
8-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
9-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←
A-		!	"	#	$	%	&	'	(	)	*	+	'	–	.	/
B-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C-	–	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
D-	p	q	r	s	t	u	v	w	x	y	z	—	—	—	—	—
E-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
F-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

(Righe 8–C:) Questi caratteri vengono visualizzati in inverso

FIXCAR

Come abbiamo visto, i codici schermo del VIC richiedono una trasformazione dall'ASCII. Tuttavia FIXCAR non dovrà solo eseguire una conversione di codice ma anche settare le appropriate locazioni della memoria colore. Il codice sorgente presentato qui di seguito eseguirà quanto richiesto:

```
FIXCAR                      Il carattere è nell'accumulatore, ma pri-
                            ma di convertirlo è necessario settare il
                            byte della memoria colore con il codice
                            colore stabilito
        PHA                 Salva il carattere da visualizzare
        LDA TV.PTR+1
        PHA
        CLC                 TV.PTR deve puntare al byte corrispon-
        ADC #$84            dente nella memoria colore
        STA TV.PTR+1
        LDY #0
        LDA $286            Legge il codice colore
        STA (TV.PTR),Y
        PLA                 Resetta il byte alto di TV.PTR
        STA TV.PTR+1
        PLA                 Recupera il carattere da visualizzare
        SEC
        CMP #$40            È minore di $40?
        BCC FINFIX          Se sì, non è necessario convertire
        CMP #$60
        BCC SUB.40          Se è nell'intervallo $40-$5F sottrae $40
                            per convertire
        SBC #$20            Sottrae $20 per convertire (minuscolo)
        RTS
SUB.40  SEC
        SBC #$40            Sottrae $40 per convertire (maiuscolo)
FINFIX  RTS                 L'accumulatore ora contiene il codice
                            VIC
```

Routine di input da tastiera

Per leggere un carattere ASCII dalla tastiera del VIC potremo usare la seguente subroutine:

```
VICKEY  JSR $FFE4     Locazione di ROM della routine di scan-
                      sione della tastiera
        CMP #0        Se è 0 non è stato premuto alcun tasto
        BEQ VICKEY
        RTS           Restituisce il carattere ASCII dalla ta-
                      stiera
```

Questa routine riporta il codice ASCII maiuscolo di ogni tasto premuto e il codice ASCII di ogni interpunzione o numero.

VICTVT

Per visualizzare un carattere ASCII sullo schermo sarà sufficiente chiamare la routine in ROM alla locazione $FFD2 a cui ci riferiremo col nome di VICTVT. Ogni carattere ASCII visualizzabile sarà correttamente stampato alla corrente locazione indicata da VICTVT. È possibile cambiare la corrente locazione di TVT (che non è lo stesso delle utility di schermo del Capitolo 5) richiamando VICTVT con l'accumulatore settato con uno dei codici di controllo della Tabella B1.2.

Tabella B1.2 Codici di controllo del VIC-20

Carattere	Codice	Funzione
CURSORE SU	$91	Muove il cursore in su di una riga
CURSORE DESTRA	$1D	Muove il cursore a destra di una posizione
CURSORE GIÙ	$11	Muove il cursore in giù di una riga
CURSORE SINISTRA	$9D	Muove il cursore a sinistra di una posizione
INSERIMENTO	$94	Muove il carattere selezionato, e tutti i caratteri alla sua destra, a destra di una posizione
CANCELLAZIONE	$14	Muove il carattere selezionato e tutti i caratteri alla sua destra, a sinistra di una posizione
HOME	$13	Setta la corrente locazione all'angolo superiore sinistro dello schermo
CLEAR	$93	Setta la corrente locazione all'angolo superiore sinistro dello schermo e pulisce il video
REVERSE	$12	Visualizza i caratteri che seguono in inverso
REVERSE-OFF	$92	Visualizza i caratteri che seguono in normale

Questi codici di controllo possono essere passati direttamente a VICTVT oppure essere inclusi nei testi delle stringhe da visualizzare. Per esempio, per pulire lo schermo prima di visualizzare un messaggio basta inserire all'inizio della stringa da visualizzare il carattere CLEAR ($93).
Con questo sistema, per visualizzare il messaggio sulla quinta riga dello schermo basterà inserire dopo il codice CLEAR quattro codici CURSORE GIÙ ($11, $11, $11, $11).
Può darsi che i codici di controllo non vi serviranno mai, ma è buona cosa sapere che esistono.

Riservare la memoria

Prima di caricare in memoria il Monitor Visibile o le sue estensioni bisogna assicurarsi che l'interprete BASIC del VIC non faccia uso dell'area di memoria sopra alla locazione $2FFF. Per riservare questa area di memoria basterà digitare le seguenti istruzioni subito dopo l'accensione del computer:

```
POKE 52,47 : POKE 56,47: POKE 51,255 : POKE 55,255
NEW
```

(Non dimenticate di premere RETURN dopo l'inserimento di ciascuna linea).

Dopo aver digitato queste istruzioni sarà possibile memorizzare qualsiasi programma in BASIC senza alterare l'area di memoria superiore a $2FFF.
Ricordate che è necessario compiere questa operazione prima di caricare il Monitor Visibile (vedi Capitolo 13).

Attivazione del Monitor Visibile

Dopo aver caricato il Monitor Visibile (usando il LOADER e la serie delle Appendici E) sarà possibile attivarlo da BASIC con il comando:

```
SYS 12431
```

mentre per tornare al BASIC basterà digitare Q (da Quit — uscire).

Come ottenere la stampa

Le routine di stampa esadecimale, di stampa del disassembler e del text editor indirizzano l'uscita ad una stampante. Attualmente però questo non è vero perché queste routine indirizzano l'uscita alla periferica che avete definito come file logico #2 sia esso una stampante, un disk drive, un modem o qualsiasi altra periferica.
Per indirizzare le stampe del software presentato in questo libro a una periferica bisogna prima di tutto attivare la periferica come file logico #2. Il modo più semplice per compiere questa operazione è eseguire un breve programma BASIC prima di attivare il Monitor Visibile. Per esempio, il programma riportato qui di seguito apre un canale a 1200 baud sulla porta seriale e poi passa il controllo al Monitor Visibile.

```
100 OPEN 2,2,0,CHR$(8)
110 SYS 12431
120 CLOSE 2
```

La linea 100 attiva la porta seriale RS232 come file logico #2. La linea 110 trasferisce il controllo al Monitor Visibile (vedi Appendice C13). Dal Monitor Visibile sarà poi possibile selezionare qualsiasi routine di stampa e l'output sarà indirizzato alla porta seriale. Se alla porta seriale è collegata una stampante si otterrà la stampa del dump, del disassemblato o del testo.
Cambiando la linea 100 con una istruzione che apre un file su disco si otterrà la memorizzazione permanente dell'output.
È quindi possibile indirizzare l'output dei programmi a ogni periferica semplicemente attivandola come file logico #2 prima di richiamare il Monitor Visibile.
NOTA: se nessuna periferica viene attivata come file logico #2, potrà verificarsi il caso in cui, chiamando una delle routine di stampa, i caratteri appariranno doppi sul video. Il rimedio in questi casi è semplice; basterà tornare al BASIC e attivare la periferica desiderata come file logico #2 prima di ritornare al Monitor Visibile.

B.2 Il C-64

Il Commodore 64 è un computer molto sofisticato pur restando nella categoria economica. A differenza di molti altri sistemi nella stessa categoria di prezzo il C-64 possiede ben 64K di RAM; molto più di quello che ci serve per poter usare il software presentato in questo libro.
La memoria video è normalmente mappata a partire dalla locazione $400. Lo schermo può contenere 25 righe di 40 caratteri ciascuna. L'indirizzo di ogni locazione di schermo è 40 ($28) byte maggiore della posizione schermo immediatamente al di sopra. Quindi i parametri schermo del C-64 saranno:

```
HOME     .WORD $400
RIGINC   .BYTE 40     Differenza di indirizzo tra una riga e la
                      successiva
TVCOL    .BYTE 39
TVRIG    .BYTE 24
```

Questo però non è tutto ciò che ci occorre sapere circa lo schermo del C-64.
In alcuni computer è possibile visualizzare ogni carattere desiderato, semplicemente memorizzando il suo codice ASCII in una data locazione sullo schermo. Visualizzare caratteri ASCII sul C-64 è leggermente più complicato. Prima di tutto c'è da considerare la memoria del colore, poi c'è il problema di determinare l'appropriato codice schermo del C-64 per un dato carattere.

Memoria colore

Oltre alla memoria video il C-64 riserva anche una zona di memoria chiamata *memoria colore*. Ogni byte nella memoria video ha un corrispondente nella memoria colore. La memoria video specifica il carattere da visualizzare e, naturalmente, la memoria colore specifica il colore con cui visualizzare quel carattere. Nel C-64 la memoria colore si trova a partire dalla locazione $D800 cioè $D400 byte sopra la memoria video, quindi l'n-esimo byte della memoria colore definisce il colore dell'n-esimo carattere della memoria video.
Notate che se una parte della memoria colore contiene il codice del colore di sfondo, nessun carattere sarà visibile nella corrispondente porzione dello schermo, anche se esiste del testo in questa porzione. Quindi, prima di visualizzare un carattere sarà necessario memorizzare l'appropriato codice colore nella memoria colore.

Che colore dovremo usare? Potremo usare un colore arbitrario, per esempio il nero, ma sarebbe più sensato usare il colore selezionato dall'utente. La locazione $286 (646 decimale) contiene il codice colore selezionato dall'utente, quindi prima di visualizzare un carattere leggeremo il codice colore nella locazione $286 e memorizzeremo questo codice all'appropriata locazione nella memoria colore. Questa locazione sarà esattamente $D400 byte più in alto della locazione a cui punta TV.PTR. Sappiamo ora come predisporre la memoria colore quando desideriamo visualizzare del testo sullo schermo, ma, come fare a determinare gli appropriati codici schermo da usare?

Codici schermo

Per visualizzare un dato carattere sullo schermo bisogna memorizzare il codice appropriato in memoria. La tabella B2.1 mostra i *codici schermo* del C-64. In questa tabella tutti i caratteri grafici sono rappresentati da un trattino, ma se volete visualizzarli sarà sufficiente eseguire il programma seguente:

```
100 REM VISUALIZZA I CODICI SCHERMO DEL C-64
110 REM IN UNA MATRICE 16×16
120 REM
130 PRINT CHR$(147) : REM PULISCE LO SCHERMO
140 VIDEO=1024  : REM MEMORIA VIDEO
150 CMEM=55296 : REM MEMORIA COLORE
160 C=PEEK(646) : REM COLORE SELEZIONATO
170 REM
180 FOR RIG=0 TO 15
190 FOR COL=0 TO 15
200 POKE CMEM+COL+40*RIG,C
210 POKE VIDEO+COL+40*RIG,COL+16*RIG
220 NEXT COL,RIG
230 REM
240 GOTO 240
```

Questo programma pulisce lo schermo, memorizza il codice colore selezionato nella memoria colore e inserisce tutti i 256 codici di schermo in una griglia 16 per 16 della memoria video. Fatto questo continuerà in un loop senza fine alla linea 240. Per uscirne sarà sufficiente premere il tasto RUN/STOP.

Tabella B2.1 Set di caratteri del C-64

	Byte basso																
Byte alto	**-0**	**-1**	**-2**	**-3**	**-4**	**-5**	**-6**	**-7**	**-8**	**-9**	**-A**	**-B**	**-C**	**-D**	**-E**	**-F**	
0-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←	
2-		!	"	#	$	%	&	'	(	)	*	+	'	-	.	/	
3-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
4-	—	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
5-	p	q	r	s	t	u	v	w	x	y	z	—	—	—	—	—	
6-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
7-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
8-	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	Questi caratteri vengono visualizzati in inverso (righe 8-D)
9-	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←	
A-		!	"	#	$	%	&	'	(	)	*	+	'	-	.	/	
B-	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
C-	—	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
D-	p	q	r	s	t	u	v	w	x	y	z	—	—	—	—	—	
E-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	
F-	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	

FIXCAR

Come abbiamo visto, i codici schermo del C-64 richiedono una trasformazione dall'ASCII. Tuttavia FIXCAR non dovrà solo eseguire una conversione di codice ma anche settare le appropriate locazioni della memoria colore. Il codice sorgente presentato qui di seguito eseguirà quanto richiesto:

```
FIXCAR                       Il carattere è nell'accumulatore, ma pri-
                             ma di convertirlo è necessario settare il
                             byte della memoria colore con il codice
                             colore stabilito
         PHA                 Salva il carattere da visualizzare
         LDA TV.PTR+1
         PHA
         CLC                 TV.PTR deve puntare al byte corrispon-
         ADC #$D4            dente nella memoria colore
         STA TV.PTR+1
         LDY #0
```

```
        LDA $286        Legge il codice colore
        STA (TV.PTR),Y
        PLA             Resetta il byte alto di TV.PTR
        STA TV.PTR+1
        PLA             Recupera il carattere da visualizzare
        SEC
        CMP #$40        È minore di $40
        BCC FINFIX      Se sì, non è necessario convertire
        CMP #$60
        BCC SUB.40      Se è nell'intervallo $40–$5F sottrae $40
                        per convertire
        SBC #$20        Sottrae $20 per convertire (minuscolo)
        RTS
SUB.40  SEC
        SBC #$40        Sottrae $40 per convertire (maiuscolo)
FINFIX  RTS             L'accumulatore ora contiene il codice
                        C-64
```

Routine di input da tastiera

Per leggere un carattere ASCII dalla tastiera del C-64 potremo usare la seguente subroutine:

```
C64KEY  JSR $FFE4       Locazione di ROM della routine di scan-
                        sione della tastiera
        CMP #0          Se è 0 non è stato premuto alcun tasto
        BEQ C64KEY
        RTS             Restituisce il carattere ASCII dalla tastiera
```

Questa routine riporta il codice ASCII maiuscolo di ogni tasto premuto e il codice ASCII di ogni interpunzione o numero.

C64TVT

Per visualizzare un carattere ASCII sullo schermo sarà sufficiente chiamare la routine in ROM alla locazione $FFD2 a cui ci riferiremo col nome di C64TVT. Ogni carattere ASCII visualizzabile sarà correttamente

Tabella B2.2 Codici di controllo del C-64

Carattere	Codice	Funzione
CURSORE SU	$91	Muove il cursore in su di una riga
CURSORE DESTRA	$1D	Muove il cursore a destra di una posizione
CURSORE GIÙ	$11	Muove il cursore in giù di una riga
CURSORE SINISTRA	$9D	Muove il cursore a sinistra di una posizione
INSERIMENTO	$94	Muove il carattere selezionato e tutti i caratteri alla sua destra, a destra di una posizione
CANCELLAZIONE	$14	Muove il carattere selezionato e tutti i caratteri alla sua destra, a sinistra di una posizione
HOME	$13	Setta la corrente locazione all'angolo superiore sinistro dello schermo
CLEAR	$93	Setta la corrente locazione all'angolo superiore sinistro dello schermo e pulisce il video
REVERSE	$12	Visualizza i caratteri che seguono in inverso
REVERSE-OFF	$92	Visualizza i caratteri che seguono in normale

stampato alla corrente locazione indicata da C64TVT. È possibile cambiare la corrente locazione di TVT (che non è lo stesso delle utility di schermo del Capitolo 5) richiamando C64TVT con l'accumulatore settato con uno dei codici di controllo della Tabella B2.2.
Questi codici di controllo possono essere passati direttamente a C64TVT oppure essere inclusi nei testi delle stringhe da visualizzare. Per esempio, per pulire lo schermo prima di visualizzare un messaggio basta inserire all'inizio della stringa da visualizzare il carattere CLEAR ($93).
Con questo sistema, per visualizzare il messaggio sulla quinta riga dello schermo basterà inserire dopo il codice CLEAR quattro codici CURSORE GIÙ ($11, $11, $11, $11).
Può darsi che i codici di controllo non vi serviranno mai, ma è buona cosa sapere che esistono.

Riservare la memoria

Prima di caricare in memoria il Monitor Visibile o le sue estensioni bisogna assicurarsi che l'interprete BASIC del C-64 non faccia uso dell'area di memoria sopra alla locazione $2FFF. Per riservare questa area di memoria basterà digitare le seguenti istruzioni subito dopo l'accensione del computer:

```
POKE 52,47 : POKE 56,47: POKE 51,255 : POKE 55,255
NEW
```

(Non dimenticate di premere RETURN dopo l'inserimento di ciascuna linea).

Dopo aver digitato queste istruzioni sarà possibile memorizzare qualsiasi programma in BASIC senza alterare l'area di memoria superiore a $2FFF.
Ricordate che è necessario compiere questa operazione prima di caricare il Monitor Visibile (vedi Capitolo 13).

Attivazione del Monitor Visibile

Dopo aver caricato il Monitor Visibile (usando il LOADER e la serie delle Appendici E) sarà possibile attivarlo da BASIC con il comando:

```
SYS 12431
```

mentre per tornare al BASIC basterà digitare Q (da Quit — uscire).

Come ottenere la stampa

Le routine di stampa esadecimale, di stampa del disassembler e del text editor indirizzano l'uscita ad una stampante. Attualmente però questo non è vero perché queste routine indirizzano l'uscita alla periferica che avete definito come file logico #2 sia essa una stampante, un disk drive, un modem o qualsiasi altra periferica.
Per indirizzare le stampe del software presentato in questo libro a una periferica bisogna prima di tutto attivare la periferica come file logico #2. Il modo più semplice per compiere questa operazione è eseguire un breve programma BASIC prima di attivare il Monitor Visibile. Per esempio, il programma riportato qui di seguito apre un canale a 1200 baud sulla porta seriale e poi passa il controllo al Monitor Visibile.

```
100 OPEN 2,2,0,CHR$(8)
110 SYS 12431
120 CLOSE 2
```

La linea 100 attiva la porta seriale RS232 come file logico #2. La linea 110 trasferisce il controllo al Monitor Visibile (vedi Appendice C13). Dal Monitor Visibile sarà poi possibile selezionare qualsiasi routine di stampa e l'output sarà indirizzato alla porta seriale. Se alla porta seriale è

collegata una stampante si otterrà la stampa del dump, del disassemblato o del testo.
Cambiando la linea 100 con una istruzione che apre un file su disco si otterrà la memorizzazione permanente dell'output.
È quindi possibile indirizzare l'output dei programmi a ogni periferica semplicemente attivandola come file logico #2 prima di richiamare il Monitor Visibile.
NOTA: se nessuna periferica viene attivata come file logico #2, potrà verificarsi il caso in cui, chiamando una delle routine di stampa, i caratteri appariranno doppi sul video. Il rimedio in questi casi è semplice; basterà tornare al BASIC e attivare la periferica desiderata come file logico #2 prima di ritornare al Monitor Visibile.

Appendice

Listati delle routine C

C.1 Utility per la gestione dello schermo

```
1000                    ;       APPENDICE C1: UTILITY PER LA GESTIONE
1010                    ;                  DELLO SCHERMO
1020                    ;
1030                    ;
1040                    ;
1050                    ;                VEDI CAPITOLO 5
1060                    ;
1070                    ;
1080                    ;
1090                    ;
1100                    ;
1110                    ;
1120                    ;         COPYRIGHT (C) 1983 DI K. SKIER
1130                    ;           LEXINGTON, MASSACHUSETTS
1140                    ;
1150                    ;
1160                    ;
1170                    ;
1180                    ;
1190                    ;
1200                    ;
1210                    ;
1220                    ;
1230                    ;
1240                    ;
1250                    ; ****************************************
1260                    ;
1270                    ;             BYTE DI PAGINA ZERO
1280                    ;
1290                    ; ****************************************
1300                    ;
1310                    ;
1320                    ;
1330                    ;
1340                    ;
```

```
1350          TV.PTR = $FB   QUESTO PUNTATORE CONTIENE
1360    ;                    L'INDIRIZZO DELL'ATTUALE
1370    ;                    POSIZIONE SULLO SCHERMO.
1380    ;
1390    ;
1400    ;
1410    ;
1420    ;
1430    ;
1440    ;
1450    ;
1460    ; *******************************************
1470    ;
1480    ;               PARAMETRI SCHERMO
1490    ;
1500    ; *******************************************
1510    ;
1520    ;
1530    ;
1540          PARAMS = $3000
1550    ;                    I SEGUENTI INDIRIZZI DEVO-
1560    ;                    NO ESSERE INIZIALIZZATI IN
1570    ;                    MODO DA CONTENERE I DATI
1580    ;                    SCHERMO DEL VOSTRO SISTEMA
1590    ;
1600    ;
1610    ;
1620    ;
1630          HOME   = PARAMS
1640    ;                    HOME E' UN PUNTATORE CHE
1650    ;                    DEFINISCE L'ANGOLO
1660    ;                    SUPERIORE SINISTRO
1665    ;                    DELLO SCHERMO.
1670          RIGINC = PARAMS+2
1680    ;                    RIGINC E' UN BYTE CHE CON-
1690    ;                    TIENE LA DIFFERENZA TRA
1700    ;                    GLI INDIRIZZI DI UNA RIGA
1710    ;                    CON LA SUCCESSIVA.
1720          TVCOL  = PARAMS+3
1730    ;                    TVCOL E' UN BYTE CHE CON-
1740    ;                    TIENE IL NUMERO DI COLONNE
1750    ;                    (CONTANDO DA ZERO).
1760    ;
1770          TVRIG  = PARAMS+4
1780    ;                    TVRIG E' UN BYTE CHE CON-
1790    ;                    TIENE IL NUMERO DI RIGHE
1800    ;                    (CONTANDO DA ZERO).
1810    ;
1820          HIPAGE = PARAMS+5
1830    ;                    HIPAGE E' IL BYTE ALTO
1840    ;                    DEL PIU' ALTO INDIRIZZO
1850    ;                    SULLO SCHERMO.
1860    ;
1870          NULL   = PARAMS+6
1880    ;                    CODICE DEL CARATTERE
1890    ;                    SPAZIO SUL SISTEMA.
1900    ;
1910          FRECC  = PARAMS+7
1920    ;                    CODICE DEL CARATTERE
1930    ;                    FRECCIA VERSO L'ALTO
1935    ;                    SUL SISTEMA.
1940    ;
1950          FIXCAR = PARAMS+$11
1960    ;                    FIXCAR E'LA SUBROUTINE CHE
1970    ;                    CONVERTE L'ASCII NEL CODI-
1980    ;                    CE USATO DAL VIDEO DEL
```

```
1990                         ;                    VOSTRO SISTEMA.
2000                         ;
2010                         ;
2020                         ;
2030                         ;
2040                         ;
2050   0000  = 3100                    * =  $3100
2060                         ;
2070                         ;
2080                         ;
2090                         ;
2100                         ;
2110                         ;
2120                         ;
2130                         ;
2140                         ;  ***************************************
2150                         ;
2160                         ;              PULISCE LO SCHERMO
2170                         ;
2180                         ;  ***************************************
2190                         ;
2200                         ;
2210                         ;
2220                         ;
2230                         ;
2240                         ;
2250                         ;       PULISCE LO SCHERMO SENZA ALTERARE LA
2260                         ;                  PAGINA ZERO.
2270                         ;
2280                         ;
2290                         ;
2300   3100  20C431         PULTV   JSR TVSALV     SALVA I BYTE DI PAGINA
2310                         ;                     ZERO CHE SARANNO CAMBIATI.
2320   3103  202B31                 JSR TVHOME     SI POSIZIONA NELL'ANGOLO
2330                         ;                     SUPERIORE SINISTRO
2335                         ;                     DEL VIDEO.
2340   3106  AE0330                 LDX TVCOL      X E Y CONTENGONO LE DIMEN-
2350   3109  AC0430                 LDY TVRIG      SIONI X,Y DELLO SCHERMO.
2360   310C  201331                 JSR PULXY      PULISCE X COLONNE E
2370                         ;                     Y RIGHE DALLA CORRENTE
2375                         ;                     POSIZIONE.
2380   310F  20D331                 JSR TVREC      RESETTA I BYTE DI PAGINA
2390                         ;                     ZERO CHE SONO CAMBIATI.
2400   3112  60                     RTS            RITORNA CON I BYTE DI PA-
2410                         ;                     GINA ZERO INALTERATI.
2420                         ;
2430                         ;
2440                         ;
2450                         ;
2460                         ;
2470                         ;
2480                         ;
2490                         ;
2500                         ;
2510                         ;
2520                         ;
2530                         ;  ***************************************
2540                         ;
2550                         ;        PULISCE UNA PARTE DELLO SCHERMO
2560                         ;
2570                         ;  ***************************************
2580                         ;
2590                         ;
2600                         ;
2610                         ;
2620                         ;                     PULISCE X COLONNE
```

```
2630                        ;                        E Y RIGHE
2635                                                 DALLA POSIZIONE ATTUALE.
2640                        ;                        SPOSTA TV.PTR DI Y RIGHE
2650                        ;
2660                        ;
2670                        ;
2680   3113  8E2A31         PULXY  STX COL           NUMERO DI COLONNE
2690                        ;                        DA PULIRE.
2700   3116  98                    TYA
2710   3117  AA                    TAX               X CONTIENE IL NUMERO DI
2720                        ;                        COLONNE DA PULIRE.
2730                        ;
2740   3118  AD0630         PULRIG LDA NULL          LE PULISCE SCRIVENDO DE-
2750                        ;                        GLI SPAZI SULLO SCHERMO.
2760                        ;
2770   311B  AC2A31                LDY COL           Y E' IL NUMERO DI COLONNE
2780                        ;                        DA PULIRE
2790   311E  91FB           PULPOS STA (TV.PTR),Y PULISCE UNA POSIZIONE
2800                        ;                        SCRIVENDO UNO SPAZIO.
2810                        ;
2820   3120  88                    DEY               AGGIORNA L'INDICE PER LA
2830                        ;                        PROSSIMA POSIZIONE.
2840                        ;
2850   3121  10FB                  BPL PULPOS        SE NON E' FINE RIGA,
2860                        ;                        PULISCE LA POSIZIONE
2870                        ;                        SUCCESSIVA
2880   3123  207631                JSR TVGIU         SE FINE RIGA SI SPOSTA
2890                        ;                        VERSO IL BASSO DI UNA RIGA
2900                        ;
2910                        ;
2920   3126  CA                    DEX               E' L'ULTIMA RIGA ?
2930   3127  10EF                  BPL PULRIG        SE NO,PULISCE LA PROSSIMA
2940   3129  60                    RTS               SE SI', RITORNA.
2950                        ;
2960   312A   00            COL    .BYTE 0           BYTE CHE  CONTIENE  IL
2970                        ;                        NUMERO DI COLONNE
2980                        ;                        DA PULIRE
2990                        ;
3000                        ;
3010                        ;
3020                        ;
3030                        ;
3040                        ;
3050                        ;
3060                        ;
3070                        ;
3080                        ; ******************************************
3090                        ;
3100                        ;                      TVHOME
3110                        ;
3120                        ; ******************************************
3130                        ;
3140                        ;
3150                        ;
3160                        ;
3170                        ;
3180   312B  A200           TVHOME LDX #0            SETTA TV.PTR ALL'ANGOLO
3190   312D  A000                  LDY #0            SUPERIORE SINISTRO AZZE-
3200                        ;                        RANDO I REGISTRI X E Y
3210   312F  18                    CLC
3220   3130  900A                  BCC TVTOXY
3230                        ;
3240                        ;
3250                        ;
3260                        ;
3270                        ; ******************************************
```

```
3280                     ;
3290                     ;                          CENTRO
3300                     ;
3310                     ; ******************************************
3320                     ;
3330                     ;
3340                     ;
3350                     ;
3360                     ;                     SETTA TV.PTR AL CENTRO
3370                     ;                     DELLO SCHERMO
3380                     ;
3390                     ;
3400                     ;
3410                     ;
3420  3132  AD0430      CENTRO LDA TVRIG      A E' IL NUMERO DI RIGHE
3430  3135  4A                 LSR A         DIVIDE A PER DUE.
3440  3136  A8                 TAY           Y E' IL NUMERO DELLA RIGA
3450                     ;                     CENTRALE DELLO SCHERMO.
3460                     ;
3470  3137  AD0330             LDA TVCOL     A E' IL NUMERO TOTALE
3475                     ;                     DI COLONNE.
3480  313A  4A                 LSR A         DIVIDE A PER DUE.
3490  313B  AA                 TAX           X E' IL NUMERO DELLA
3500                     ;                     COLONNA CENTRALE
3510                     ;                     DELLO SCHERMO.
3520                     ;
3530                     ;                     X E Y CONTENGONO LE COOR-
3540                     ;                     DINATE X,Y DEL CENTRO
3550                     ;                     DELLO SCHERMO.
3560                     ;
3570                     ;
3580                     ;
3590                     ;
3600                     ;
3610                     ;
3620                     ;
3630                     ;
3640                     ;
3650                     ;
3660                     ;
3670                     ;
3680                     ;
3690                     ; ******************************************
3700                     ;
3710                     ;                     TVTOXY
3720                     ;
3730                     ; ******************************************
3740                     ;
3750                     ;
3760                     ;
3770                     ;
3780                     ;
3790  313C  38          TVTOXY  SEC           SETTA LA POSIZIONE
3800                     ;                     CORRENTE ALLE COORDINATE
3810                     ;                     SPECIFICATE DAI REGISTRI
3820                     ;                     X E Y.
3830  313D  EC0330              CPX TVCOL     X FUORI SCALA ?
3840  3140  9003                BCC X.OK      SE NO, VA BENE COSI'.
3850                     ;                     SE SI' ASSEGNA A X IL MAS-
3860  3142  AE0330              LDX TVCOL     SIMO VALORE AMMESSO.
3870                     ;
3880                     ;
3890  3145  38          X.OK    SEC           Y FUORI SCALA ?
3900  3146  CC0430              CPY TVRIG
3910  3149  9003                BCC Y.OK      SE NO, VA BENE COSI'.
3920                     ;
```

```
3930                          ;                         SE SI' ASSEGNA A Y IL MAS-
3940   314B  AC0430                   LDY TVRIG         SIMO VALORE AMMESSO.
3950                          ;
3960                          ;
3970                          ;
3980   314E  AD0030           Y.OK    LDA HOME          SETTA TV.PTR ALL'INDIRIZ-
3990   3151  85FB                     STA TV.PTR        ZO DELLA PARTE PIU' BASSA
4000   3153  AD0130                   LDA HOME+1        DELLO SCHERMO
4010   3156  85FC                     STA TV.PTR+1
4020                          ;
4030   3158  08                       PHP               SALVA IL FLAG DECIMALE.
4040   3159  D8                       CLD
4050                          ;
4060                          ;
4070   315A  8A                       TXA               SOMMA X A TV.PTR
4080   315B  18                       CLC
4090   315C  65FB                     ADC TV.PTR
4100   315E  9003                     BCC SETCOL
4110   3160  E6FC                     INC TV.PTR+1
4120   3162  18                       CLC
4130                          ;
4140                          ;
4150   3163  C000             SETCOL  CPY #0            SOMMA TV.PTR A Y*RIGINC
4160   3165  F00B                     BEQ FINE
4170   3167  18               LOOP    CLC
4180   3168  6D0230                   ADC RIGINC
4190   316B  9002                     BCC *+4
4200   316D  E6FC                     INC TV.PTR+1
4210   316F  88                       DEY
4220   3170  D0F5                     BNE LOOP
4230                          ;
4240                          ;
4250   3172  85FB             FINE    STA TV.PTR
4260   3174  28                       PLP               RIPORTA IL FLAG DECIMALE
4270   3175  60                       RTS               RITORNA ...
4280                          ;
4290                          ;
4300                          ;
4310                          ;
4320                          ;
4330                          ;
4340                          ;
4350                          ;
4360                          ;
4370                          ;
4380                          ;  *******************************************
4390                          ;
4400                          ;           TVGIU, TVVAI, E TVPIU
4410                          ;
4420                          ;  *******************************************
4430                          ;
4440                          ;
4450                          ;
4460                          ;
4470                          ;
4480   3176  AD0230           TVGIU LDA RIGINC        SPOSTA TV.PTR DI UNA RIGA
4490   3179  18                       CLC               IN BASSO
4500   317A  9005                     BCC TVPIU
4510                          ;
4520   317C  209B31           VEDCAR JSR TV.SCR         SCRIVE UN CARATTERE SULLO
4530                          ;                         SCHERMO E
4540                          ;
4550   317F  A901             TVVAI LDA #1              SALTA UNA POSIZIONE
4560                          ;                         INCREMENTANDO TV.PTR
4570                          ;
4580                          ;
```

```
4590   3181  08            TVPIU PHP              TVPIU SOMMA L'ACCUMULATORE
4600   3182  D8                  CLD              A TV.PTR, MANTENENDOLO
4610   3183  18                  CLC              NELLA MEMORIA VIDEO.
4620   3184  65FB                ADC TV.PTR
4630   3186  9002                BCC *+4
4640   3188  E6FC                INC TV.PTR+1
4650   318A  85FB                STA TV.PTR
4660   318C  38                  SEC              LA POSIZIONE ATTUALE E'
4670   318D  AD0530              LDA HIPAGE       FUORI DALLA MEMORIA VIDEO?
4680   3190  C5FC                CMP TV.PTR+1
4690   3192  B005                BCS TV.OK
4700                       ;
4710   3194  AD0130              LDA HOME+1       SE SI', TORNA ALLA
4720   3197  85FC                STA TV.PTR+1     POSIZIONE HOME.
4730                       ;
4740   3199  28            TV.OK  PLP
4750   319A  60                  RTS
4760                       ;
4770                       ;
4780                       ;
4790                       ;
4800                       ;
4810                       ;
4820                       ;
4830                       ;
4840                       ;
4850                       ;
4860                       ; ******************************************
4870                       ;
4880                       ;                    TV.SCR
4890                       ;
4900                       ; ******************************************
4910                       ;
4920                       ;
4930                       ;
4940                       ;
4950                       ;
4960                       ;
4970   319B  201130        TV.SCR JSR FIXCAR      CONVERTE L'ASCII NEL
4980                       ;                      CODICE DEL SISTEMA.
4990                       ;
5000                       ;
5010   319E  A000                LDY #0           SCRIVE IL CARATTERE NELLA
5020   31A0  91FB                STA (TV.PTR),Y CORRENTE LOCAZIONE
5030   31A2  60                  RTS              E RITORNA.
5040                       ;
5050                       ;
5060                       ;
5070                       ;
5080                       ;
5090                       ;
5100                       ;
5110                       ;
5120                       ;
5130                       ; ******************************************
5140                       ;
5150                       ;     VISUALIZZA UN BYTE IN FORMATO ESA
5160                       ;
5170                       ; ******************************************
5180                       ;
5190                       ;
5200                       ;
5210                       ;
5220                       ;
5230   31A3  48            VEDBYT PHA             SALVA IL BYTE
5235                       ;                      DA VISUALIZZARE.
```

```
5240   31A4  4A               LSR A         MUOVE IL MSB NELLA POSI-
5250   31A5  4A               LSR A         ZIONE OCCUPATA DAI 4 LSB.
5260   31A6  4A               LSR A
5270   31A7  4A               LSR A
5280                  ;
5290   31A8  20B631           JSR ASCII     DETERMINA IL CARATTERE
5300                  ;                     ASCII DEI 4 LSB DI A.
5310                  ;
5320   31AB  207C31           JSR VEDCAR    VISUALIZZA IL CARATTERE E
5330                  ;                     AVANZA ALLA PROSSIMA
5340                  ;                     LOCAZIONE.
5350                  ;
5360   31AE  68               PLA           RIPORTA IL BYTE ORIGINALE
5370   31AF  20B631           JSR ASCII     IN A E DETERMINA L'ASCII
5380                  ;                     DEI 4 LSB.
5390                  ;
5400   31B2  207C31           JSR VEDCAR    SALVA QUESTO CARATTERE ASCII
5410                  ;                     ALLA DESTRA DEL PRECEDEN-
5420                  ;                     TE E AVANZA ALLA
5430                  ;                     SUCCESSIVA POSIZIONE.
5440                  ;
5450                  ;
5460   31B5  60               RTS           RITORNA ...
5470                  ;
5480                  ;
5490                  ;
5500                  ;
5510                  ;
5520                  ;
5530                  ;
5540                  ;
5550                  ;
5560                  ;
5570                  ; ********************************************
5580                  ;
5590                  ;                    DA ESA A ASCII
5600                  ;
5610                  ; ********************************************
5620                  ;
5630                  ;
5640                  ;
5650                  ;
5660                  ;
5670   31B6  08       ASCII   PHP           DETERMINA L'ASCII DEI 4
5680   31B7  D8               CLD           LSB DI A.
5690   31B8  290F             AND #$0F      AZZERA I 4 MSB.
5700   31BA  C90A             CMP #$0A      L'ACCUMULATORE E' > 9?
5710                  ;
5720   31BC  3002             BMI DECIML    SE NO, E' COMPRESO TRA
5730                  ;                     0 E 9.
5740   31BE  6906             ADC #6        SE SI', SARA' TRA A E F.
5750                  ;                     SOMMA $36 PER
5760                  ;                     LA CONVERSIONE.
5770   31C0  6930     DECIML  ADC #$30      SE A E' TRA 0 E 9
5780                  ;                     SOMMA $30 PER LA
5790                  ;                     CONVERSIONE.
5800                  ;
5810   31C2  28               PLP           RESETTA IL FLAG DECIMALE
5820                  ;
5830   31C3  60               RTS           RITORNA...
5840                  ;
5850                  ;
5860                  ;
5870                  ;
5880                  ;
5890                  ;
```

```
5900                    ;
5910                    ;
5920                    ;
5930                    ;
5940                    ;
5950                    ;
5960                    ;
5970                    ; *********************************************
5980                    ;
5990                    ;                   TVSALV
6000                    ;
6010                    ; *********************************************
6020                    ;
6030                    ;
6040                    ;
6050                    ;                     SALVA LA POSIZIONE COR-
6060                    ;                     RENTE NELLO STACK.
6070                    ;
6080                    ;
6090                    ;
6100                    ;
6110                    ;
6120  31C4  68          TVSALV PLA            LEGGE L'INDIRIZZO DI RI-
6130  31C5  AA                 TAX            TORNO E LO SALVA IN X E Y
6140  31C6  68                 PLA
6150  31C7  A8                 TAY
6160                    ;
6170                    ;
6180  31C8  A5FC               LDA TV.PTR+1   LEGGE TV.PTR E LO SALVA
6190  31CA  48                 PHA            NELLO STACK.
6200  31CB  A5FB               LDA TV.PTR
6210  31CD  48                 PHA
6220                    ;
6230                    ;
6240  31CE  98                 TYA            RIPORTA NELLO STACK
6250  31CF  48                 PHA            L'INDIRIZZO DI RITORNO.
6260  31D0  8A                 TXA
6270  31D1  48                 PHA
6280                    ;
6290                    ;
6300  31D2  60                 RTS            RITORNA CON TV.PTR NELLO
6310                    ;                     STACK CON IL BYTE BASSO
6320                    ;                     IN CIMA.
6330                    ;
6340                    ;
6350                    ;
6360                    ;
6370                    ;
6380                    ;
6390                    ;
6400                    ;
6410                    ;
6420                    ;
6430                    ; *********************************************
6440                    ;
6450                    ;                   TVREC
6460                    ;
6470                    ; *********************************************
6480                    ;
6490                    ;
6500                    ;
6510                    ;                     RESETTA LA POSIZIONE
6520                    ;                     SULLO SCHERMO LEGGENDOLA
6530                    ;                     NELLO STACK.
6540                    ;
6550                    ;
```

```
6560   31D3  68         TVREC  PLA             LEVA DALLO STACK L'INDI-
6570   31D4  AA                TAX             RIZZO DI RITORNO E LO
6580   31D5  68                PLA             SALVA IN X E Y.
6590   31D6  A8                TAY
6600                    ;
6610                    ;
6620   31D7  68                PLA             RESETTA TV.PTR DALLO
6630   31D8  85FB              STA TV.PTR      STACK.
6640   31DA  68                PLA
6650   31DB  85FC              STA TV.PTR+1
6660                    ;
6670                    ;
6680   31DD  98                TYA             RIPORTA NELLO STACK
6690   31DE  48                PHA             L'INDIRIZZO DI RITORNO.
6700   31DF  8A                TXA
6710   31E0  48                PHA
6720                    ;
6730                    ;
6740   31E1  60                RTS             RITORNA...
```

CROSS REFERENCE

```
LOOP    3167    FRECC   3007    ASCII   31B6    NULL    3006
CENTRO  3132    PULTV   3100    PULXY   3113    PULPOS  311E
PULRIG  3118    COL     312A    SETCOL  3163    DECIML  31C0
FIXCAR  3011    HIPAGE  3005    HOME    3000    PARAMS  3000
RIGINC  3002    TV.OK   3199    TVREC   31D3    TV.PTR  00FB
TV.SCR  319B    FINE    3172    TVCOL   3003    TVGIU   3176
TVHOME  312B    TVPIU   3181    TVSALV  31C4    TVRIG   3004
TVVAI   317F    TVTOXY  313C    VEDBYT  31A3    VEDCAR  317C
X.OK    3145    Y.OK    314E
```

C.2 Il Monitor Visibile
(primo livello e subroutine di visualizzazione)

```
1000   ;        APPENDICE C2: LISTATO ASSEMBLER DEL
1010   ;                      MONITOR VISIBILE
1020   ;
1030   ;         PRIMO LIVELLO E SUBROUTINE VIDEO
1040   ;
1050   ;
1060   ;
1070   ;
1080   ;
1090   ;                VEDI CAPITOLO 6
1100   ;
1110   ;
1120   ;
1130   ;
1140   ;
1150   ;             COPYRIGHT (C) 1983 DI K. SKIER
1160   ;                LEXINGTON, MASSACHUSETTS
1170   ;
1180   ;
1190   ;
1200   ;
1210   ;
1220   ;
1230   ;
1240   ;
1250   ;
1260   ;
1270   ;
1280   ;
1290   ;
1300   ;
1310   ;
1320   ;
1330   ; ********************************************
1340   ;
1350   ;                  UGUAGLIANZE
1360   ;
1370   ; ********************************************
1380   ;
1390   ;
1400   ;
1410   ;
1420          TV.PTR = $FB
1430   ;
1440          LEGPTR = $FB
1450   ;
1460   ;
1470          PARAMS = $3000   INDIRIZZO DEI DATI DI
1480   ;                       SISTEMA.
1490   ;
1500          TVCOL  = PARAMS+3
1510   ;                       TVCOL E' UN BYTE CHE CON-
1520   ;                       TIENE IL NUMERO DI COLONNE
1530   ;                       (CONTANDO DA ZERO).
1540   ;
1550   ;
1560          FRECC  = PARAMS+7
1570   ;                       CODICE DEL CARATTERE
1580   ;                       FRECCIA VERSO L'ALTO
1590   ;                       SUL SISTEMA.
```

```
1600                        ;
1610                             ROMKEY = PARAMS+8
1620                        ;                      ROMKEY PUNTA ALLA SUBROU-
1630                        ;                      TINE DI SISTEMA CHE
1640                        ;                      LEGGE UN CARATTERE ASCII
1650                        ;                      DALLA TASTIERA.
1660                        ;
1670                             SPAZIO = $20
1680                        ;
1690                             RUBOUT = $7F
1700                        ;
1710                             CR     = $0D      ASCII PER 'A CAPO'.
1720                        ;
1730                        ;
1740                        ;
1750                        ;
1760                        ;
1770                        ;
1780                        ;
1790                        ;
1800                        ;
1810                        ;
1820                        ;
1830                        ;
1840                        ; ********************************************
1850                        ;
1860                        ;         SUBROUTINE NECESSARIE
1870                        ;
1880                        ; ********************************************
1890                        ;
1900                        ;
1910                        ;
1920                             TVSUBS = $3100
1930                             PULTV  = TVSUBS
1940                             PULXY  = TVSUBS+$13
1950                             TVHOME = TVSUBS+$2B
1960                             TVTOXY = TVSUBS+$3C
1970                             TVGIU  = TVSUBS+$76
1980                             VEDCAR = TVSUBS+$7C
1990                             TVVAI  = TVSUBS+$7F
2000                             TVPIU  = TVSUBS+$81
2010                             VEDBYT = TVSUBS+$A3
2020                             ASCII  = TVSUBS+$B6
2030                             TVSALV = TVSUBS+$C4
2040                             TVREC  = TVSUBS+$D3
2050                        ;
2060                        ;
2070                        ;
2080   0000   = 3200            * = $3200
2090                        ;
2100                        ;
2110                        ;
2120                             AGGIOR = *+$E3
2130                        ;
2140                        ;
2150                        ;
2160                        ;
2170                        ;
2180                        ;
2190                        ;
2200                        ;
2210                        ;
2220                        ;
2230                        ;
2240                        ;
2250                        ;
```

```
2260                    ; ********************************************
2270                    ;
2280                    ;          AREA MODIFICABILE DALL'UTENTE
2290                    ;
2300                    ; ********************************************
2310                    ;
2320                    ;
2330                    ;
2340                    ;
2350                    ;
2360   3200  00         CAMPO  .BYTE 0       NUMERO DEL CAMPO SCELTO.
2370                    ;                    (DEVE ESSERE TRA 0 E 6)
2380                    ;
2390   3201  00         REG.A  .BYTE 0       IMMAGINE DI A.
2400                    ;
2410   3202  00         REG.X  .BYTE 0       IMMAGINE DI X.
2420                    ;
2430   3203  00         REG.Y  .BYTE 0       IMMAGINE DI Y.
2440                    ;
2450   3204  00         REG.P  .BYTE 0       IMMAGINE DI P.
2460                    ;
2470                    ;
2480                           REGI   = REG.A
2490                    ;
2500   3205  0000       SELEZ  .WORD 0       PUNTATORE DELL'ATTUALE
2510                    ;                    INDIRIZZO.
2520                    ;
2530                    ;
2540                    ;
2550                    ;
2560                    ;
2570                    ;
2580                    ;
2590                    ;
2600                    ; ********************************************
2610                    ;
2620                    ;              IL MONITOR VISIBILE
2630                    ;
2640                    ; ********************************************
2650                    ;
2660                    ;
2670                    ;
2680                    ;
2690   3207  08         VISMON PHP             SALVA IL FLAG DI STATO.
2700   3208  D8                CLD             AZZERA IL MODO DECIMALE
2710                    ;                      PERCHE' LE OPERAZIONI IN
2720                    ;                      QUESTO LIBRO SONO BINARIE.
2730                    ;
2740   3209  201232            JSR VISUAL      VISUALIZZA IL MONITOR.
2750                    ;
2760                    ;
2770   320C  20E332            JSR AGGIOR      LEGGE LA RICHIESTA
2780                    ;                      DELL'UTENTE.
2790   320F  18                CLC
2800   3210  90F6              BCC VISMON+1
2810                    ;
2820                    ;
2830                    ;
2840                    ;
2850                    ;
2860                    ;
2870                    ;
2880                    ;
2890                    ;
2900                    ; ********************************************
2910                    ;
```

```
2920                          ;            VIDEATA DEL MONITOR
2930                          ;
2940                          ; ******************************************
2950                          ;
2960                          ;
2970                          ;
2980                          ;
2990                          ;
3000   3212  20C431          VISUAL JSR TVSALV    SALVA I BYTE DI PAGINA
3010                          ;                   ZERO CHE SARANNO ALTERATI
3020                          ;
3030   3215  202532                 JSR PULMON    PULISCE UNA PARTE DEL
3040   3218  203532                 JSR LINE.1    VIDEO E MOSTRA LE LINEE
3050   321B  205D32                 JSR LINE.2    DEI TITOLO DEI DATI E
3060   321E  20B032                 JSR LINE.3    INDICE.
3070                          ;
3080   3221  20D331                 JSR TVREC     RESETTA I BYTE DI PAGINA
3090                          ;                   ZERO.
3100                          ;
3110   3224  60                     RTS           RITORNA.
3120                          ;
3130                          ;
3140                          ;
3150                          ;
3160                          ;
3170                          ;
3180                          ;
3190                          ;
3200                          ;
3210                          ;
3220                          ; ******************************************
3230                          ;
3240                          ;        PULISCE PARTE DELLO SCHERMO
3250                          ;
3260                          ; ******************************************
3270                          ;
3280                          ;
3290                          ;
3300                          ;
3310                          ;
3320   3225  A200            PULMON LDX #0        SETTA TV.PTR ALLA POSI-
3330   3227  A000                   LDY #0        ZIONE HOME.
3340   3229  203C31                 JSR TVTOXY
3350                          ;
3360   322C  AE0330                 LDX TVCOL     X E' IL NUMERO DI COLONNE
3370                          ;                   DA PULIRE.
3380                          ;
3390   322F  A003                   LDY #3        Y E' IL NUMERO DI RIGHE
3400                          ;                   DA PULIRE (3).
3410                          ;
3420   3231  201331                 JSR PULXY     PULISCE X COLONNE,Y RIGHE
3430                          ;
3440   3234  60                     RTS           RITORNA...
3450                          ;
3460                          ;
3470                          ;
3480                          ;
3490                          ;
3500                          ;
3510                          ;
3520                          ;
3530                          ;
3540                          ;
3550                          ; ******************************************
3560                          ;
3570                          ;          MOSTRA LA LINEA TITOLI
```

```
3580                         ;
3590                         ; ******************************************
3600                         ;
3610                         ;
3620                         ;
3630                         ;
3640                         ;
3650  3235  A20B             LINE.1 LDX #11       COORDINATA X DI "A".
3660  3237  A000                    LDY #0        COORDINATA Y DI "A".
3670  3239   203C31                 JSR  TVTOXY   SETTA TV.PTR ALLA LOCA-
3680                         ;                    ZIONE DI SCHERMO DI "A".
3690                         ;
3700  323C  A000                    LDY #0        VISUALIZZA I TITOLI
3710  323E  8C5232                  STY TITCOL    INIZIALIZZA IL CONTATORE
3720                         ;                    DELLA COLONNA DEI TITOLI.
3730                         ;
3740  3241  B95332           TITLOP LDA TITOLI,Y  PRENDE UN·CARATTERE E LO
3750  3244  207C31                  JSR VEDCAR    VISUALIZZA SULLO SCHERMO.
3760  3247  EE5232                  INC TITCOL    PREPARA PER IL PROSSIMO
3765                         ;                    CARATTERE.
3770  324A  AC5232                  LDY TITCOL    E' L'ULTIMO?
3780  324D  C00A                    CPY #10
3790  324F   D0F0                   BNE  TITLOP   SE NO,VISUALIZZA IL PROS-
3800                         ;                    SIMO.
3810  3251  60                      RTS           RITORNA ...
3820  3252  00               TITCOL .BYTE 0       LOCAZIONE: CONTIENE I CA-
3830                         ;                    RATTERI DA RICOPIARE.
3840                         ;
3850                         ;
3860                         ;
3870                         ;
3880  3253  4120205820       TITOLI .BYTE 'A  X  Y  P'     ;
3890        2059202050
3900                         ;
3910                         ;
3920                         ;
3930                         ;
3940                         ;
3950                         ;
3960                         ;
3970                         ;
3980                         ;
3990                         ;
4000                         ; ******************************************
4010                         ;
4020                         ;              MOSTRA LA LINEA DATI
4030                         ;
4040                         ; ******************************************
4050                         ;
4060                         ;
4070                         ;
4080                         ;
4090                         ;
4100  325D  A200             LINE.2 LDX #0        X E' LA COLONNA DI
4110                         ;                    PARTENZA DELLA LINEA DATI
4120                         ;
4130  325F  A001                    LDY #1        Y E' IL NUMERO DI RIGA
4140                         ;                    DELLA LINEA DATI.
4150                         ;
4160  3261  203C31                  JSR TVTOXY    SETTA  TV.PTR ALL'INIZIO
4170                         ;                    DELLA LINEA DATI.
4180                         ;
4190  3264  AD0632                  LDA SELEZ +1  VISUALIZZA IL BYTE ALTO
4200  3267  20A331                  JSR VEDBYT    DELL'INDIRIZZO.
4210  326A  AD0532                  LDA SELEZ     VISUALIZZA IL BYTE BASSO
4220  326D  20A331                  JSR VEDBYT    DELL'INDIRIZZO.
```

```
4230                    ;
4240  3270  207F31      ;        JSR TVVAI       SALTA UNO SPAZIO DOPO IL
4250                    ;                        CAMPO DI INDIRIZZO.
4260                    ;
4270  3273  209532               JSR LEGBYT      LEGGE IL BYTE SELEZIONATO
4280                    ;
4290                    ;
4300  3276  48                   PHA             E LO SALVA NELLO STACK.
4310                    ;
4320  3277  20A331               JSR VEDBYT      LO VISUALIZZA NEL CAMPO 1,
4330                    ;                        IN FORMATO ESADECIMALE.
4340                    ;
4350  327A  207F31               JSR TVVAI       SALTA UNO SPAZIO DOPO
4360                    ;                        IL CAMPO 1.
4370                    ;
4380  327D  68                   PLA             RIPORTA IL CORRENTE BYTE
4390                    ;                        IN A.
4400                    ;
4410  327E  207C31               JSR VEDCAR      LO VISUALIZZA NEL CAMPO 2
4420                    ;                        COME CARATTERE.
4430                    ;
4440  3281  207F31               JSR TVVAI       SALTA UNO SPAZIO DOPO IL
4450                    ;                        CAMPO 2.
4460                    ;
4470                    ;                        VISUALIZZA IL CONTENUTO
4480                    ;                        DEI REGISTRI NEI CAMPI
4490                    ;                        DAL 3 AL 6:
4500  3284  A200                 LDX #0          INIZIA CON L'IMMAGINE DI
4510                    ;                        A.
4520  3286  BD0132      VEDREG   LDA REGI,X      VISUALIZZA IL REGISTRO
4530  3289  20A331               JSR VEDBYT      IN FORMATO ESADECIMALE.
4540  328C  207F31               JSR TVVAI       SALTA UNO SPAZIO.
4550                    ;
4560                    ;
4570  328F  E8                   INX             PREPARA PER IL PROSSIMO
4575                    ;                        REGISTRO.
4580  3290  E004                 CPX #4          E' IL QUARTO ?
4590  3292  D0F2                 BNE  VEDREG     SE NO, PROSEGUE.
4600                    ;
4610  3294  60                   RTS             SE SI', RITORNA ...
4620                    ;
4630                    ;
4640                    ;
4650                    ;
4660                    ;
4670                    ;
4680                    ;
4690                    ;
4700                    ;
4710                    ;
4720                    ;
4730                    ;
4740                    ; *****************************************
4750                    ;
4760                    ;            LEGGE IL BYTE SELEZIONATO
4770                    ;
4780                    ; *****************************************
4790                    ;
4800                    ;
4810                    ;
4820                    ;
4830                    ;
4840                    ;
4850  3295  A5FB        LEGBYT   LDA LEGPTR      LEGGE IL BYTE SELEZIONA-
4860  3297  48                   PHA             TO DAL PUNTATORE (CONSER-
4870  3298  A6FC                 LDX LEGPTR+1    VANDO LA PAGINA ZERO)
```

```
4880                        ;
4890   329A   AD0532                LDA SELEZ
4900   329D   85FB                  STA LEGPTR
4910   329F   AD0632                LDA SELEZ +1
4920   32A2   85FC                  STA LEGPTR+1
4930                        ;
4940   32A4   A000                  LDY #0
4950   32A6   B1FB                  LDA (LEGPTR),Y
4960   32A8   A8                    TAY
4970   32A9   68                    PLA
4980   32AA   85FB                  STA LEGPTR
4990   32AC   86FC                  STX LEGPTR+1
5000   32AE   98                    TYA
5010   32AF   60                    RTS             RITORNA ...
5020                        ;
5030                        ;
5040                        ;
5050                        ;
5060                        ;
5070                        ;
5080                        ;
5090                        ;
5100                        ;
5110                        ;
5120                        ; ****************************************
5130                        ;
5140                        ;             MOSTRA LA LINEA INDICE
5150                        ;
5160                        ; ****************************************
5170                        ;
5180                        ;
5190                        ;
5200                        ;
5210                        ;
5220   32B0   AC0032        LINE.3 LDY CAMPO        LEGGE IL CAMPO CORRENTE
5230   32B3   38                   SEC
5240   32B4   C007                 CPY #7
5250   32B6   9005                 BCC CAM.OK
5260   32B8   A000                 LDY #0
5270   32BA   8C0032               STY CAMPO
5280   32BD   B9CD32        CAM.OK LDA CAMPI ,Y     E IL NUMERO DI COLONNA
5290                        ;                       PER IL CORRENTE CAMPO
5300   32C0   AA                   TAX              CHE SARA' LA COORDINATA X
5310                        ;                       DELL'INDICE
5320   32C1   A002                 LDY #2           SETTA LA COORDINATA Y.
5330   32C3   203C31               JSR  TVTOXY      TV.PTR PUNTA ALLA
5340                        ;                       LOCAZIONE DELL'INDICE
5350                        ;
5360   32C6   AD0730               LDA FRECC        METTE UNA FRECCIA
5370   32C9   207C31               JSR VEDCAR       IN QUELLA LOCAZIONE
5380   32CC   60                   RTS              E RITORNA ...
5390                        ;
5400                        ;
5410   32CD   030608        CAMPI  .BYTE 3,6,8      QUEST'AREA DEFINISCE LE
5420   32D0   0B0E                 .BYTE $0B,$0E    COLONNE CHE POSSONO CON-
5430   32D2   1114                 .BYTE $11,$14    TENERE LA FRECCIA INDICE
5440                        ;
5450                        ;                       VARIANDO QUESTI VALORI,
5460                        ;                       LA FRECCIA APPARIRA'
5470                        ;                       IN COLONNE DIFFERENTI.
5480                        ;
5490                        ;
5500                        ;
```

CROSS REFERENCE

FRECC	3007	ASCII	31B6	PULTV	3100	PULXY	3113
PULMON	3225	CR	000D	VISUAL	3212	CAMPO	3200
CAMPI	32CD	CAM.OK	32BD	LEGBYT	3295	LEGPTR	00FB
TITOLI	3253	TITCOL	3252	TITLOP	3241	LINE.1	3235
LINE.2	325D	LINE.3	32B0	PARAMS	3000	REG.A	3201
REG.P	3204	REG.X	3202	REG.Y	3203	REGI	3201
ROMKEY	3008	RUBOUT	007F	SELEZ	3205	SPACE	0020
TVREC	31D3	TV.PTR	00FB	TVCOL	3003	TVGIU	3176
TVHOME	312B	TVPIU	3181	TVSALV	31C4	TVVAI	317F
TVSUBS	3100	TVTOXY	313C	AGGIOR	32E3	VISMON	3207
VEDBYT	31A3	VEDCAR	317C	VEDREG	3286		

C.3 Il Monitor Visibile (subroutine di aggiornamento)

```
1000                    ;   APPENDICE C3: LISTATO ASSEMBLER DEL
1010                    ;                 MONITOR VISIBILE
1020                    ;
1030                    ;
1040                    ;                 SUBROUTINE AGGIOR
1050                    ;
1060                    ;
1070                    ;
1080                    ;                  VEDI CAPITOLO 6
1090                    ;
1100                    ;
1110                    ;
1120                    ;
1130                    ;
1140                    ;           COPYRIGHT (C) 1983 DI K. SKIER
1150                    ;             LEXINGTON, MASSACHUSETTS
1160                    ;
1170                    ;
1180                    ;
1190                    ;
1200                    ;
1210                    ;
1220                    ;
1230                    ;
1240                    ;
1250                    ;
1260                    ;
1270                    ; ********************************************
1280                    ;
1290                    ;                  UGUAGLIANZE
1300                    ;
1310                    ; ********************************************
1320                    ;
1330                    ;
1340                    ;
1350                    ;
1360                    ;
1370                    ;
1380                    ;
1390                        TV.PTR = $FB
1400                    ;
1410                    ;
1420                        PARAMS = $3000  INDIRIZZO DEI DATI DI
1430                    ;                   SISTEMA
1440                    ;
1450                    ;
1460                        FRECC  = PARAMS+7 CODICE DI SISTEMA
1470                    ;                     EQUIVALENTE A FRECCIA
1480                    ;                     VERSO L'ALTO
1490                    ;
1500                    ;
1510                        ROMKEY = PARAMS+8 PUNTATORE ALLA ROUTINE
1520                    ;                     DI SISTEMA PER
1530                    ;                     L'ACQUISIZIONE DI UN
1540                    ;                     CARATTERE ASCII DALLA
1550                    ;                     TASTIERA
1560                    ;
1570                        COMOD  = PARAMS+$10 ROUTINE FITTIZIA:
1580                    ;                       NON ESEGUE NULLA
1590                    ;
```

```
1600                    ;
1610                    ;
1620                        SPAZIO = $20
1630                    ;
1640                        CLRKEY = 147   TASTO 'CLR'
1650                    ;
1660                        CR     = $0D   ASCII PER 'A CAPO'
1670                    ;
1680                    ;
1690                    ;
1700                    ;
1710                    ;
1720                    ;
1730                    ;
1740                    ;
1750                    ;
1760                    ;
1770                    ;
1780                    ;
1790                    ; ****************************************
1800                    ;
1810                    ;          SUBROUTINE NECESSARIE
1820                    ;
1830                    ; ****************************************
1840                    ;
1850                    ;
1860                    ;
1870                    ;
1880                    ;
1890                        TVSUBS = $3100
1900
1910                        PUL.TV = TVSUBS  PULISCE LO SCHERMO
1920                    ;
1930                    ;
1940                    ;
1950                        VMSUBS = $3200   PUNTO D'INIZIO DEL
1960                    ;                    CODICE DEL MONITOR
1970                    ;                    VISIBILE
1980                    ;
1990                        LEGBYT = VMSUBS+$95 ACQUISISCE IL BYTE
2000                    ;                       ATTUALMENTE SELE-
2010                    ;                       ZIONATO
2020                    ;
2030                    ;
2040                    ;
2050                    ;
2060                    ;
2070                    ;
2080                    ;
2090                    ;
2100                    ;
2110                    ;
2120                    ;
2130                    ;
2140                    ;
2150                    ; ****************************************
2160                    ;
2170                    ;      AREA MODIFICABILE DALL'UTENTE
2180                    ;
2190                    ; ****************************************
2200                    ;
2210                    ;
2220                    ;
2230                    ;
2240                    ;
2250                        CAMPO  = VMSUBS        NUMERO DEL CAMPO
```

```
2260                      ;                       SCELTO (DEVE ESSE-
2270                      ;                       RE TRA 0 E 6).
2280                           REG.A  = VMSUBS+1  IMMAGINE DI A.
2290                      ;
2300                           REG.X  = VMSUBS+2  IMMAGINE DI X.
2310                      ;
2320                           REG.Y  = VMSUBS+3  IMMAGINE DI Y.
2330                      ;
2340                           REG.P  = VMSUBS+4  IMMAGINE DI P.
2350                      ;
2360                      ;
2370                           REGI = REG.A
2380                      ;
2390                           SELEZ  = VMSUBS+5  PUNTATORE DELL'AT-
2400                      ;                       TUALE INDIRIZZO.
2410                      ;
2420                      ;
2430                      ;
2440                      ;
2450                      ;
2460                      ;
2470                      ;
2480                      ;
2490                      ;
2500                      ;
2510                      ; ****************************************
2520                      ;
2530                      ;           ROUTINE DI INPUT
2540                      ;
2550                      ; ****************************************
2560                      ;
2570                      ;
2580  0000  = 32E0               * = VMSUBS+$E0
2590                      ;
2600                      ;
2610  32E0  6C0830        LEGKEY JMP (ROMKEY)   JSR LEGKEY RICHIAMA IN
2620                      ;                     MODO INDIRETTO LA ROU-
2630                      ;                     TINE DI INPUT DALLA TA-
2640                      ;                     STIERA.
2650                      ;
2660                      ;
2670                      ;
2680                      ;
2690                      ;
2700                      ;
2710                      ;
2720                      ;
2730                      ;
2740                      ;
2750                      ; ****************************************
2760                      ;
2770                      ;                 AGGIOR
2780                      ;
2790                      ; ****************************************
2800                      ;
2810                      ;
2820                      ;
2830                      ;
2840                      ;
2850  32E3  20E032        AGGIOR JSR LEGKEY     LEGGE UN CARATTERE DALLA
2860                      ;                     TASTIERA
2870                      ;
2880  32E6  C91D                 CMP #$1D       E' IL TASTO '->' ?
2890  32E8  D010                 BNE IF.LFT     SE NO, ESEGUE IL TEST
2900                      ;                     SUCCESSIVO.
2910  32EA  EE0032        NEXT.F INC CAMPO      SE SI' SELEZIONA IL CAM-
```

```
2920   32ED  AD0032             LDA CAMPO      PO SUCCESSIVO.
2930   32F0  C907               CMP #7         SE L'INDICE ERA AL CAMPO
2940   32F2  D005               BNE VIA 1      PIU' A DESTRA,SI POSIZIO-
2950   32F4  A900               LDA #0         NA SOTTO QUELLO PIU' A
2960   32F6  8D0032             STA CAMPO      SINISTRA
2970   32F9  60          VIA 1  RTS            E RITORNA
2980                      ;
2990                      ;
3000   32FA  C99D        IF.LFT CMP #$9D       E' IL TASTO '<-' ?
3010   32FC  D00B               BNE IF.SP      SE NO ESEGUE IL TEST
3020                      ;                    SUCCESSIVO
3030   32FE  CE0032      PREV.F DEC CAMPO      SE SI' SELEZIONA IL CAMPO
3040   3301  1005               BPL VIA 2      PRECEDENTE:SE L'INDICE
3050   3303  A906               LDA #6         ERA AL CAMPO PIU'A SINI-
3060   3305  8D0032             STA CAMPO      STRA SI POSIZIONA SOTTO
3070   3308  60          VIA 2  RTS            QUELLO PIU' A DESTRA E
3080                      ;                    RITORNA.
3090                      ;
3100   3309  C920        IF.SP CMP #SPAZIO     E' UNO SPAZIO?
3110   330B  D009               BNE IF.CR      SE NO ESEGUE IL TEST
3120                      ;                    SUCCESSIVO.
3130   330D  EE0532      INC.SL INC SELEZ      SE SI' AVANZA IN MEMORIA
3140   3310  D003               BNE *+5        INCREMENTANDO IL PUNTATO-
3150   3312  EE0632             INC SELEZ +1   RE CHE SPECIFICA L'INDI-
3160                      ;                    RIZZO VISUALIZZATO
3170   3315  60                 RTS            E RITORNA
3180                      ;
3190                      ;
3200   3316  C90D        IF.CR  CMP #CR        E' UN 'A CAPO'?
3210   3318  D00C               BNE IFCAR      SE NO, ESEGUE IL TEST
3220                      ;                    SUCCESSIVO.
3230   331A  AD0532      DEC.SL LDA SELEZ      SE SI' RETROCEDE IN MEMO-
3240   331D  D003               BNE *+5        RIA DECREMENTANDO IL PUN-
3250   331F  CE0632             DEC SELEZ +1   TATORE CHE SPECIFICA
3260   3322  CE0532             DEC SELEZ      L'INDIRIZZO VISUALIZZATO
3270   3325  60                 RTS            E RITORNA.
3280                      ;
3290                      ;
3300   3326  AE0032      IFCAR  LDX CAMPO      L'INDICE E' SOTTO IL
3310   3329  E002               CPX #2         CAMPO 2?
3320   332B  D01B               BNE IF.GO      SE NO, ESEGUE IL TEST
3330                      ;                    SUCCESSIVO
3340   332D  A8          MEM.SL TAY            SE SI' MEMORIZZA IL CA-
3350   332E  A5FB               LDA TV.PTR     RATTERE ALL'INDIRIZZO
3360   3330  48                 PHA            SELEZIONATO (CONSERVAN-
3370   3331  A6FC               LDX TV.PTR+1   DO LA PAGINA ZERO)
3380   3333  AD0532             LDA SELEZ
3390   3336  85FB               STA TV.PTR
3400   3338  AD0632             LDA SELEZ +1
3410   333B  85FC               STA TV.PTR+1
3420   333D  98                 TYA
3430   333E  A000               LDY #0
3440   3340  91FB               STA (TV.PTR),Y
3450   3342  86FC               STX TV.PTR+1
3460   3344  68                 PLA
3470   3345  85FB               STA TV.PTR
3480   3347  60                 RTS            E RITORNA.
3490                      ;
3500                      ;
3510   3348  C947        IF.GO  CMP #'G'       E' IL TASTO 'G'??
3520   334A  D023               BNE IF.HEX     SE NO, ESEGUE IL  TEST
3530                      ;                    SUCCESSIVO.
3540   334C  AC0332      GO     LDY REG.Y      SE SI' CARICA I REGI-
3550   334F  AE0232             LDX REG.X      STRI DEL 6502 CON LA
3560   3352  AD0432             LDA REG.P      LORO IMMAGINE
3570   3355  48                 PHA            VISUALIZZATA.
```

```
3580   3356  AD0132          LDA REG.A
3590   3359  28              PLP
3600   335A  206C33          JSR CALLSL      CHIAMA LA SUBROUTINE AL-
3610   335D  08              PHP             L'INDIRIZZO SELEZIONATO
3620   335E  8D0132          STA REG.A       E AL RITORNO SALVA IL
3630   3361  8E0232          STX REG.X       VALORE DEI REGISTRI NEL-
3640   3364  8C0332          STY REG.Y       LE RISPETTIVE IMMAGINI.
3650   3367  68              PLA
3660   3368  8D0432          STA REG.P
3670   336B  60              RTS             E RITORNA.
3680                  ;
3690                  ;
3700   336C  6C0532   CALLSL JMP (SELEZ )    JSR CALLSL CHIAMA IN MODO
3710                  ;                      INDIRETTO L'INDIRIZZO
3720                  ;                      SELEZIONATO.
3730                  ;
3740                  ;
3750   336F  48       IF.HEX PHA             SALVA IL CARATTERE.
3760   3370  20D533          JSR BINA        SE A CONTIENE UN CARAT-
3770                  ;                      TERE ASCII COMPRESO TRA
3780                  ;                      0 - 9 O TRA A - F LO
3790                  ;                      CONVERTE IN BINARIO.
3800   3373  304B            BMI IF.CLR      SE NO ESEGUE IL TEST
3810                  ;                      SUCCESSIVO.
3820                  ;
3830                  ;
3840   3375  A8              TAY             PRENDE UN CARATTERE DAL-
3850   3376  68              PLA             LO STACK E NE SALVA L'E-
3860   3377  98              TYA             QUIVALENTE BINARIO IN A
3870                  ;                      E Y.
3880   3378  AE0032          LDX CAMPO       L'INDICE E' SOTTO IL
3890   337B  D014            BNE NOINDI      CAMPO 0? SE NO DEVE ES-
3900                  ;                      SERE SOTTO UN ALTRO
3905                  ;                      CAMPO ESADECIMALE
3910   337D  A203     CAMPO1 LDX #3          POICHE' L'INDICE E' SOTTO
3920   337F  18       LOOP.1 CLC             AL CAMPO INDIRIZZI METTE
3930   3380  0E0532          ASL SELEZ       IL CARATTERE ESADECIMALE
3940   3383  2E0632          ROL SELEZ +1    NEL CAMPO INDIRIZZI,
3950   3386  CA              DEX             SETTANDO IL PUNTATORE CHE
3960   3387  10F6            BPL LOOP.1      SPECIFICA L'INDIRIZZO VI-
3970   3389  98              TYA             SUALIZZATO.
3980   338A  0D0532          ORA SELEZ
3990   338D  8D0532          STA SELEZ
4000   3390  60              RTS             E RITORNA.
4010                  ;
4020                  ;
4030   3391  E001     NOINDI CPX #1          E' SOTTO IL CAMPO 1?
4040   3393  D018            BNE CAMPOR      SE NO DEVE ESSERE SOTTO
4050                  ;                      ALL'IMMAGINE DI UN
4060                  ;                      REGISTRO
4070   3395  290F     ROLLSL AND #$0F        SPOSTA I 4 LSB IN A NEL
4080   3397  48              PHA             BYTE SELEZIONATO.
4090   3398  209532          JSR LEGBYT      LEGGE IL CONTENUTO DEL-
4100   339B  0A              ASL A           L'INDIRIZZO SELEZIONATO
4110   339C  0A              ASL A           E TRASLA A SINISTRA
4120   339D  0A              ASL A           4 VOLTE
4130   339E  0A              ASL A
4140   339F  29F0            AND #$F0
4150   33A1  8DAC33          STA TEMP
4160   33A4  68              PLA
4170   33A5  0DAC33          ORA TEMP
4180   33AB  202D33          JSR MEM.SL      LO METTE ALL'INDIRIZZO
4190   33AB  60              RTS             SELEZIONATO E RITORNA.
4200                  ;
4210   33AC  00       TEMP   .BYTE 0
4220                  ;
```

```
4230                     ;
4240                     ;
4250  33AD  CA           CAMPOR DEX              L'INDICE DEVE ESSERE SOT-
4260  33AE  CA                  DEX              TO AL CAMPO 3,4,5 O 6.
4270  33AF  CA                  DEX
4280  33B0  A003                LDY #3
4290                     ;
4300  33B2  18           LOOP.2 CLC              METTE IL CARATTERE ESADE-
4310  33B3  1E0132              ASL REGI,X       CIMALE NELL'APPROPRIATA
4320  33B6  88                  DEY              IMMAGINE
4330  33B7  10F9                BPL LOOP.2
4340  33B9  1D0132              ORA REGI,X
4350  33BC  9D0132              STA REGI,X
4360  33BF  60                  RTS
4370                     ;
4380                     ;
4390  33C0  68           IF.CLR PLA              RECUPERA  IL CARATTERE.
4400  33C1  C993                CMP #CLRKEY      E' IL TASTO PER PULIRE
4410                     ;                       LO SCHERMO?
4420  33C3  D004                BNE NOTCLR       SE NO, ESEGUE IL TEST
4430                     ;                       SUCCESSIVO
4440  33C5  200031              JSR PUL.TV       SE SI' PULISCE LO SCHER-
4450  33C8  60                  RTS              MO E RITORNA.
4460                     ;
4470                     ;
4480  33C9  C951         NOTCLR CMP #'Q'         E' UNA  'Q' (Q PER FINE)?
4490  33CB  D004                BNE ALTRI        SE NO, ESEGUE IL TEST
4500                     ;                       SUCCESSIVO.
4510                     ;                       SE SI' RITORNA AL
4520                     ;                       PROGRAMMA CHE HA RICHIA-
4530                     ;                       MATO VISMON
4540                     ;
4550  33CD  68                  PLA
4560  33CE  68                  PLA
4570                     ;
4580  33CF  28                  PLP              RIPORTA I FLAG INIZIALI
4590                     ;                       DEL 6502. LO STACK CON-
4600                     ;                       TIENE ORA L'INDIRIZZO DI
4610  33D0  60                  RTS              RITORNO DI VISMON PER CUI
4620                     ;                       RITORNA A VISMON.
4630                     ;
4640                     ;
4650                     ;
4660                     ;
4670                     ;
4680  33D1  201030       ALTRI  JSR COMOD        RIPORTA A COMOD CHIAMAN-
4690                     ;                       DO UNA QUALSIASI SUBROU-
4700                     ;                       TINE CHE ESTENDA LA FUN-
4710                     ;                       ZIONALITA' DEL MONITOR
4720                     ;                       VISIBILE
4730  33D4  60                  RTS              E RITORNA.
4740                     ;
4750                     ;
4760                     ;
4770                     ;
4780                     ;
4790                     ;
4800                     ;
4810                     ;
4820                     ;
4830                     ;
4840                     ;
4850                     ;
4860                     ; ******************************************
4870                     ;
4880                     ;                       BINA
```

```
4890                      ;
4900                      ;  ***************************************
4910                      ;
4920                      ;
4930                      ;
4940                      ;                     SE A CONTIENE CARATTERI
4950                      ;                     ASCII COMPRESI TRA 0 E 9
4960                      ;                     O A E F,BINA RITORNA AL-
4970                      ;                     L'EQUIVALENTE BINARIO,
4980                      ;                     ALTRIMENTI RITORNA A $FF.
4990                      ;
5000  33D5  38            BINA    SEC
5010  33D6  E930                  SBC #$30
5020  33D8  900F                  BCC NOK
5030  33DA  C90A                  CMP #$0A
5040  33DC  900E                  BCC OK
5050  33DE  E907                  SBC #7
5060  33E0  C910                  CMP #$10
5070  33E2  B005                  BCS NOK
5080  33E4  38                    SEC
5090  33E5  C90A                  CMP #$0A
5100  33E7  B003                  BCS OK
5110  33E9  A9FF          NOK     LDA #$FF
5120  33EB  60                    RTS
5130                      ;
5140  33EC  A200          OK      LDX #0
5150  33EE  60                    RTS
```

CROSS REFERENCE

```
LOOP.1  337F    CAMPO1  337D    FRECC   3007    NOK     33E9
BINA    33D5    CALLSL  336C    PUL.TV  3100    CLRKEY  0093
CR      000D    DEC.SL  331A    COMOD   3010    CAMPO   3200
LEGBYT  3295    LEGKEY  32E0    GO      334C    OK      33EC
IF.CLR  33C0    IF.CR   3316    IF.GO   3348    IF.HEX  336F
IF.LFT  32FA    IF.SP   3309    IFCAR   3326    INC.SL  330D
NEXT.F  32EA    NOINDI  3391    NOTCLR  33C9    ALTRI   33D1
PARAMS  3000    PREV.F  32FE    MEM.SL  332D    REG.A   3201
REG.P   3204    REG.X   3202    REG.Y   3203    CAMPOR  33AD
REGI    3201    LOOP.2  33B2    ROLLSL  3395    ROMKEY  3008
SELEZ   3205    SPAZI   0020    TEMP    33AC    TV.PTR  00FB
TVSUBS  3100    VIA 1   32F9    VIA 2   3308    AGGIOR  32E3
VMSUBS  3200
```

C.4 Utility di stampa

```
1000    ;     APPENDCE C4: LISTATO ASSEMBLER DELLE
1010    ;                  UTILITY DI STAMPA
1020    ;
1030    ;
1040    ;
1050    ;
1060    ;                  VEDI CAPITOLO 7
1070    ;
1080    ;
1090    ;                  DI K. SKIER
1100    ;
1110    ;
1120    ;          COPYRIGHT (C) 1983 DI K. SKIER
1130    ;             LEXINGTON, MASSACHUSETTS
1140    ;
1150    ;
1160    ;
1170    ;
1180    ;
1190    ;
1200    ;
1210    ;
1220    ;
1230    ;
1240    ;
1250    ; ********************************************
1260    ;
1270    ;                  COSTANTI
1280    ;
1290    ; ********************************************
1300    ;
1310    ;
1320    ;
1330    ;
1340          CR      = $0D   ASCII PER 'A CAPO'
1350    ;
1360          ETX     = $FF   CARATTERE DI FINE TESTO.
1370    ;
1380    ;
1390          LF      = $0A   LINE FEED
1400    ;
1410          OFF     = 0
1420    ;
1430          ON      = $FF
1440    ;
1450    ;
1460    ;
1470    ;
1480    ;
1490    ;
1500    ;
1510    ;
1520    ;
1530    ;
1540    ; ********************************************
1550    ;
1560    ;              INDIRIZZI ESTERNI
1570    ;
1580    ; ********************************************
1590    ;
1600    ;
1610    ;
```

```
1620                    ;
1630                    ;
1640                    ;
1650                    ;
1660                    ;
1670                    ;
1680                    ;
1690                          PARAMS = $3000
1700                    ;                        INDIRIZZO DEI DATI DI
1710                    ;                        SISTEMA.
1720                    ;
1730                    ;
1740                          ROMPRT = PARAMS+$0C
1750                    ;                        PUNTATORE DELLA ROUTINE
1760                    ;                        IN ROM CHE INDIRIZZA LA
1770                    ;                        PORTA SERIALE.
1780                    ;
1790                    ;
1800                          ROMTVT = PARAMS+$0A
1810                    ;                        PUNTATORE DELLA ROUTINE
1820                    ;                        IN ROM CHE INDIRIZZA LO
1830                    ;                        SCHERMO.
1840                    ;
1850                    ;
1860                          USROUT = PARAMS+$0E
1870                    ;                        PUNTATORE DELLA ROUTINE
1880                    ;                        DI OUTPUT UTENTE.
1890                    ;
1900                    ;
1910                    ;
1920                    ;
1930                          TVSUBS = $3100
1940                          ASCII  = TVSUBS+$B6
1950                    ;
1960                    ;
1970                    ;
1980                    ;
1990                          VMPAGE = $3200
2000                    ;                        PAGINA DI PARTENZA DEL
2010                    ;                        MONITOR VISIBILE.
2020                    ;
2030                          SELEZ  = VMPAGE+5
2040                          LEGBYT = VMPAGE+$95
2050                          INC.SL = VMPAGE+$10D
2060                    ;
2070                    ;
2080                    ;
2090                    ;
2100                    ;
2110                    ;
2120                    ;
2130                    ; ********************************************
2140                    ;
2150                    ;                  VARIABILI
2160                    ;
2170                    ; ********************************************
2180                    ;
2190                    ;
2200                    ;
2210                    ;
2220                    ;
2230   0000  = 3400             * = $3400
2240                    ;
2250   3400  00         PRINTR .BYTE OFF       FLAG DI USCITA STAMPANTE
2260                    ;
2270   3401  FF         TVT    .BYTE ON        FLAG DI USCITA SCHERMO
```

```
2280                    ;
2290                    ;
2300  3402  00          USR    .BYTE OFF      FLAG DI USCITA DELLA
2310                    ;                     SUBROUTINE UTENTE
2320                    ;
2330  3403  00          CAR    .BYTE 0        ULTIMO CARATTERE STAMPA-
2340                    ;                     TO DA INVCAR.
2350                    ;                     CAR =00 SIGNIFICA CHE NON
2360                    ;                     E' STATO STAMPATO ALCUN
2370                    ;                     CARATTERE.
2380                    ;
2390  3404  00          RIPETE .BYTE 0        BYTE USATO COME CONTATORE
2400                    ;                     DA CARX , SPAZI E CRLFX
2410                    ;
2420                    ;
2430                    ;
2440  3405  00          TEMP.X .BYTE 0        LOCAZIONE USATA DA PR.MSG.
2450                    ;
2460                    ;
2470  3406  0000        RETURN .WORD 0        PUNTATORE USATO DA
2480                    ;                     SAL.SL E REC.SL.
2490                    ;
2500                    ;
2510                    ;
2520                    ;
2530                    ;
2540                    ;
2550                    ;
2560                    ; ***********************************************
2570                    ;
2580                    ;      SUBROUTINE DI SELEZIONE/ESCLUSIONE
2590                    ;
2600                    ; ***********************************************
2610                    ;
2620                    ;
2630                    ;
2640                    ;
2650                    ;
2660                    ;
2670                    ;
2680  3408  A9FF        TVT.ON LDA #ON        SELEZIONA IL VIDEO COME
2690  340A  8D0134             STA TVT        PERIFERICA DI OUTPUT SET-
2700  340D  60                 RTS            TANDO IL FLAG CON IL VA-
2710                    ;                     LORE $FF.
2720                    ;
2730                    ;
2740                    ;
2750                    ;
2760                    ;
2770  340E  A900        TVTOFF LDA #OFF       DISATTIVA IL VIDEO
2780  3410  8D0134             STA TVT        AZZERANDO IL FLAG.
2790  3413  60                 RTS
2800                    ;
2810                    ;
2820                    ;
2830                    ;
2840                    ;
2850  3414  A9FF        PR.ON  LDA #ON        ATTIVA LA STAMPANTE SET-
2860  3416  8D0034             STA PRINTR     TANDO IL FLAG CON IL VA-
2870  3419  60                 RTS            LORE $FF.
2880                    ;
2890                    ;
2900                    ;
2910                    ;
2920                    ;
2930  341A  A900        PR.OFF LDA #OFF       DISATTIVA LA STAMPANTE
```

```
2940   341C  8D0034          STA PRINTR     AZZERANDO IL FLAG.
2950   341F  60              RTS
2960                  ;
2970                  ;
2980                  ;
2990                  ;
3000                  ;
3010   3420  A9FF     USR.ON LDA #ON        ATTIVA LA ROUTINE UTENTE
3020   3422  8D0234          STA USR        SETTANDO IL FLAG A $FF.
3030   3425  60              RTS
3040                  ;
3050                  ;
3060                  ;
3070                  ;
3080                  ;
3090   3426  A900     USROFF LDA #OFF       DISATTIVA LA ROUTINE
3100   3428  8D0234          STA USR        UTENTE AZZERANDO IL FLAG.
3110   342B  60              RTS
3120                  ;
3130                  ;
3140                  ;
3150                  ;
3160                  ;
3170   342C  200834   ALL.ON JSR TVT.ON     ATTIVA TUTTE LE PERI-
3180   342F  201434          JSR PR.ON      FERICHE.
3190   3432  202034          JSR USR.ON
3200   3435  60              RTS
3210                  ;
3220                  ;
3230                  ;
3240                  ;
3250                  ;
3260   3436  200E34   ALLOFF JSR TVTOFF     DISATTIVA TUTTE LE PERI-
3270   3439  201A34          JSR PR.OFF     FERICHE.
3280   343C  202634          JSR USROFF
3290   343F  60              RTS
3300                  ;
3310                  ;
3320                  ;
3330                  ;
3340                  ;
3350                  ;
3360                  ;
3370                  ;
3380                  ;
3390                  ;
3400                  ;  ******************************************
3410                  ;
3420                  ;        ROUTINE DI STAMPA DEI CARATTERI
3430                  ;
3440                  ;  ******************************************
3450                  ;
3460                  ;
3470                  ;
3480                  ;
3490                  ;
3500                  ;     STAMPA IL CARATTERE CONTENUTO IN A
3510                  ;    SU TUTTE LE PERIFERICHE SELEZIONATE.
3520                  ;
3530                  ;
3540                  ;
3550   3440  C900     INVCAR CMP #0         CONTROLLA IL CARATTERE
3560   3442  F024            BEQ FINE       SE E' UN NULL RITORNA.
3570                  ;
3580   3444  8D0334          STA CAR        SALVA IL CARATTERE.
3590                  ;
```

```
3600  3447  AD0134          LDA TVT        IL VIDEO E' ATTIVO?
3610  344A  F006            BEQ IF.PR      SE NO, ESEGUE IL TEST
3620                 ;                     SUCCESSIVO.
3630  344C  AD0334          LDA CAR        SE SI' INVIA INDIRETTA-
3640  344F  206934          JSR VAI. 1     MENTE IL CARATTERE ALLA
3650                 ;                     ROUTINE DI OUTPUT DEL
3660                 ;                     SISTEMA.
3670                 ;
3680  3452  AD0034   IF.PR  LDA PRINTR     E' ATTIVA LA STAMPANTE?
3690  3455  F006            BEQ IF.USR     SE NO, ESEGUE IL TEST
3700                 ;                     SUCCESSIVO.
3710  3457  AD0334          LDA CAR        SE SI' INVIA INDIRETTA-
3720  345A  206C34          JSR VAI. 2     MENTE IL CARATTERE ALLA
3730                 ;                     ROUTINE DI STAMPA.
3740                 ;
3750                 ;
3760  345D  AD0234   IF.USR LDA USR        E' ATTIVA LA ROUTINE
3770                 ;                     UTENTE?
3780  3460  F006            BEQ FINE       SE NO, RITORNA.
3790                 ;
3800  3462  AD0334          LDA CAR        SE SI' INVIA INDIRETTA-
3810  3465  206F34          JSR VAI. 3     MENTE IL CARATTERE ALLA
3820                 ;                     ROUTINE UTENTE.
3830                 ;
3840  3468  60       FINE   RTS            RITORNA.
3850                 ;
3860                 ;
3870                 ;
3880                 ;
3890                 ;
3900                 ;
3910                 ;
3920  3469  6C0A30   VAI. 1 JMP (ROMTVT)
3930                 ;
3940  346C  6C0C30   VAI. 2 JMP (ROMPRT)
3950                 ;
3960  346F  6C0E30   VAI. 3 JMP (USROUT)
3970                 ;
3980                 ;
3990                 ;
4000                 ;
4010                 ;
4020                 ;
4030                 ;
4040                 ; ********************************************
4050                 ;
4060                 ;   ROUTINE DI OUTPUT PER CARATTERI SPECIALI
4070                 ;
4080                 ; ********************************************
4090                 ;
4100                 ;
4110                 ;
4120                 ;
4130                 ;
4140                 ;                     CR-LF
4150                 ;
4160                 ;
4170  3472  A90D     CR.LF  LDA #CR        INVIA UN CR E LF A TUTTE
4180  3474  204034          JSR INVCAR     LE PERIFERICHE SELEZIO-
4190  3477  A90A            LDA #LF        NATE.
4200  3479  204034          JSR INVCAR
4210  347C  60              RTS            E RITORNA.
4220                 ;
4230                 ;
4240                 ;
4250                 ;
```

```
4260                      ;
4270                      ;                    SPAZIO
4280                      ;
4290                      ;
4300                      ;
4310   347D  A920         SPAZIO LDA #$20        CARICA IL CODICE ASCII
4320   347F  204034              JSR INVCAR      DELLO SPAZIO IN A E LO
4330   3482  60                  RTS             INVIA ALLE PERIFERICHE.
4340                      ;
4350                      ;
4360                      ;
4370                      ;
4380                      ;
4390                      ;
4400                      ;
4410                      ;
4420                      ;
4430                      ; ****************************************
4440                      ;
4450                      ;                    BYTE
4460                      ;
4470                      ; ****************************************
4480                      ;
4490                      ;
4500                      ;
4510                      ;
4520                      ;
4530                      ;
4540                      ;
4550                      ;  PR.BYT INVIA IL CONTENUTO DI A, IN ESADE-
4560                      ; CIMALE, A TUTTE LE PERIFERICHE SELEZIONATE.
4570                      ;
4580                      ;
4590                      ;
4600   3483  48           PR.BYT PHA             SALVA IL BYTE.
4610   3484  4A                  LSR A           DETERMINA L' ASCII CORRI-
4620   3485  4A                  LSR A           SPONDENTE AI BIT PIU'SI-
4630   3486  4A                  LSR A           GNIFICATIVI.
4640   3487  4A                  LSR A
4650   3488  20B631              JSR ASCII
4660   348B  204034              JSR INVCAR      INVIA IL CARATTERE ALLE
4670                      ;                      PERIFERICHE ATTIVE.
4680   348E  68                  PLA             DETERMINA L'ASCII DEI 4
4690   348F  20B631              JSR ASCII       BIT MENO SIGNIFICATIVI.
4700   3492  204034              JSR INVCAR      INVIA IL CARATTERE.
4710   3495  60                  RTS             RITORNA.
4720                      ;
4730                      ;
4740                      ;
4750                      ;
4760                      ;
4770                      ;
4780                      ;
4790                      ; ****************************************
4800                      ;
4810                      ;              INVIO RIPETITIVO
4820                      ;
4830                      ; ****************************************
4840                      ;
4850                      ;
4860                      ;
4870                      ;          INVIA X SPAZI
4880                      ;
4890                      ;
4900   3496  A920         SPAZI  LDA #$20        CARICA L'ASCII DEL CARAT-
4910                      ;                      TERE SPAZIO
```

```
4920                    ;                    E LO STAMPA X VOLTE:
4930                    ;
4940                    ;
4950                    ;
4960                    ;
4970                    ;       STAMPA X CARATTERI
4980                    ;
4990                    ;
5000                    ;
5010  3498  8E0434      CARX   STX RIPETE    STAMPA IL CARATTERE X
5020  349B  48          RPLOOP PHA           VOLTE E LO SALVA.
5030  349C  AE0434             LDX RIPETE    E' L'ULTIMA VOLTA?
5040  349F  F00A               BEQ FINERI    SE SI' TERMINA, SE NO
5050  34A1  CE0434             DEC RIPETE    DECREMENTA IL CONTATORE.
5060  34A4  204034             JSR INVCAR
5070                    ;
5080  34A7  68                 PLA           RIPORTA IL CARATTERE IN A
5090  34A8  18                 CLC           E LO RISTAMPA SE NECESSA-
5100  34A9  90F0               BCC RPLOOP    RIO.
5110                    ;
5120  34AB  68          FINERI PLA           PULISCE LO STACK
5130  34AC  60                 RTS           E RITORNA.
5140                    ;
5150                    ;
5160                    ;
5170                    ;             CRLFX
5180                    ;
5190                    ;
5200  34AD  8E0434      CRLFX  STX RIPETE    INIZIALIZZA IL CONTATORE.
5210  34B0  AE0434      CRLOOP LDX RIPETE    SE FINITO RITORNA.
5220  34B3  F009               BEQ FINECR
5230  34B5  CE0434             DEC RIPETE    DECREMENTA IL CONTATORE.
5240  34B8  207234             JSR CR.LF     INVIA UN 'A CAPO'
5250                    ;
5260  34BB  18                 CLC           CONTROLLA SE ERA L'ULTIMO
5270  34BC  90F2               BCC CRLOOP
5280                    ;
5290  34BE  60          FINECR RTS           SE HA FINITO RITORNA.
5300                    ;
5310                    ;
5320                    ;
5330                    ;
5340                    ;
5350                    ;
5360                    ;
5370                    ;
5380                    ;
5390                    ; ********************************************
5400                    ;
5410                    ;              PR.MSG
5420                    ;
5430                    ; ********************************************
5440                    ;
5450                    ;
5460                    ;
5470                    ;
5480                    ;
5490                    ;
5500                    ;
5510                    ;
5520  34BF  8E0534      PR.MSG STX TEMP.X    SALVA IL REGISTRO X CHE
5530                    ;                    SPECIFICA IL PUNTATORE.
5540                    ;
5550  34C2  B501               LDA 1,X       SALVA IL PUNTATORE.
5560  34C4  48                 PHA
5570  34C5  B500               LDA 0,X
```

```
5580   34C7  48                 PHA
5590                     ;
5600   34C8  AE0534      LOOP   LDX TEMP.X      RESETTA X CON IL VALORE
5610                     ;                      ORIGINALE.
5620   34CB  A100               LDA (0,X)       LEGGE IL CARATTERE SUC-
5630   34CD  C9FF               CMP #ETX        CESSIVO. E' LA FINE ?
5640   34CF  F00C               BEQ FINMSG      SE SI' SALTA A FINMSG.
5650                     ;
5660   34D1  F600               INC 0,X         SE NO,INCREMENTA IL PUN-
5670   34D3  D002               BNE  NEXT       TATORE.
5680   34D5  F601               INC 1,X
5690   34D7  204034      NEXT   JSR INVCAR      INVIA IL CARATTERE.
5700   34DA  18                 CLC
5710   34DB  90EB               BCC LOOP
5720                     ;
5730                     ;
5740   34DD  68          FINMSG PLA             RESETTA IL PUNTATORE
5750   34DE  9500               STA 0,X
5760   34E0  68                 PLA
5770   34E1  9501               STA 1,X
5780   34E3  60                 RTS             RITORNA CON MSGPTR SAL-
5790                     ;                      VATO.
5800                     ;
5810                     ;
5820                     ;
5830                     ;
5840                     ;
5850                     ;
5860                     ;
5870                     ;
5880                     ;
5890                     ;  ******************************************
5900                     ;
5910                     ;          STAMPA IL TESTO SEGUENTE
5920                     ;
5930                     ;  ******************************************
5940                     ;
5950                     ;
5960                     ;
5970                     ;
5980                     ;
5990   34E4  68          INVIA: PLA             ESTRAE L'INDIRIZZO DI RI-
6000   34E5  AA                 TAX             TORNO DALLO STACK E LO
6010   34E6  68                 PLA             SALVA IN X E Y.
6020   34E7  A8                 TAY
6030                     ;
6040   34E8  201235             JSR SAL.SL      SALVA IL PUNTATORE DI RI-
6050   34EB  8E0532             STX SELEZ       TORNO PER USARLO COME
6060   34EE  8C0632             STY SELEZ +1    PUNTATORE DI MESSAGGIO.
6070                     ;
6080                     ;
6090   34F1  200D33             JSR INC.SL      INCREMENTA SELEZ
6100                     ;
6110   34F4  200D33      NEXTCA JSR INC.SL
6120   34F7  209532             JSR LEGBYT      LEGGE IL CARATTERE.
6130   34FA  C9FF               CMP #ETX        E' LA FINE DEL TESTO?
6140   34FC  F006               BEQ STOP        SE SI', RITORNA.
6150   34FE  204034             JSR INVCAR      SE NO, LO INVIA.
6160   3501  18                 CLC             LEGGE IL CARATTERE
6170   3502  90F0               BCC NEXTCA      SUCCESSIVO
6180                     ;
6190                     ;
6200   3504  AE0532      STOP   LDX SELEZ
6210   3507  AC0632             LDY SELEZ +1
6220   350A  202B35             JSR REC.SL      RESETTA IL PUNTATORE AL
6230   350D  98                 TYA             VALORE INIZIALE E METTE
```

```
6240    350   48                PHA              L'INDIRIZZO DELL'ETX NELLO
6250    350F  8A                TXA              STACK
6260    3510  48                PHA
6270    3511  60                RTS              RITORNA AL BYTE IMMEDIATA-
6280                    ;                        MENTE DOPO L'ETX.
6290                    ;
6300                    ;
6310                    ;
6320                    ;
6330                    ;
6340                    ;
6350                    ;
6360                    ;
6370                    ;
6380                    ;
6390                    ; *******************************************
6400                    ;
6410                    ;                   SALVA SELEZ
6420                    ;
6430                    ; *******************************************
6440                    ;
6450                    ;
6460                    ;
6470                    ;
6480                    ;
6490    3512  68        SAL.SL PLA               LEGGE L'INDIRIZZO DI RI-
6500    3513  8D0634            STA RETURN       TORNO DALLO STACK E LO
6510    3516  68                PLA              METTE IN RETURN
6520    3517  8D0734            STA RETURN+1
6530                    ;
6540                    ;
6550    351A  AD0632            LDA SELEZ +1     METTE SELEZ NELLO STACK
6560    351D  48                PHA
6570    351E  AD0532            LDA SELEZ
6580    3521  48                PHA
6590                    ;
6600                    ;
6610    3522  AD0734            LDA RETURN+1     RIMETTE NELLO STACK L'IN-
6620    3525  48                PHA              DIRIZZO DI RITORNO.
6630    3526  AD0634            LDA RETURN
6640    3529  48                PHA
6650                    ;
6660                    ;
6670    352A  60                RTS              RITORNA AL PROGRAMMA
6680                    ;                        PRINCIPALE.
6690                    ;
6700                    ;
6710                    ;
6720                    ;
6730                    ;
6740                    ;
6750                    ;
6760                    ;
6770    352B  68        REC.SL PLA               SALVA L'INDIRIZZO DI
6780    352C  8D0634            STA RETURN       RITORNO.
6790    352F  68                PLA
6800    3530  8D0734            STA RETURN+1
6810                    ;
6820                    ;
6830    3533  68                PLA
6840    3534  8D0532            STA SELEZ
6850    3537  68                PLA
6860    3538  8D0632            STA SELEZ +1
6870                    ;
6880                    ;
6890    353B  AD0734    ,       LDA RETURN+1     RIMETTE L'INDIRIZZO DI
```

```
6900   353E  48              PHA           RITORNO NELLO STACK.
6910   353F  AD0634          LDA RETURN
6920   3542  48              PHA
6930                   ;
6940                   ;
6950   3543  60              RTS
6960                   ;
```

CROSS REFERENCE

```
ALL.ON 342C    ALLOFF 3436    ASCII  31B6    CAR    3403
CARX   3498    CR     000D    CR.LF  3472    CRLFX  34AD
CRLOOP 34B0    FINECR 34BE    STOP   3504    ETX    00FF
FINE   3468    LEGBYT 3295    IF.PR  3452    IF.USR 345D
INC.SL 330D    LF     000A    LOOP   34C8    FINMSG 34DD
NEXT   34D7    NEXTCA 34F4    OFF    0000    ON     00FF
PARAMS 3000    REC.SL 352B    PR.BYT 3483    INVCAR 3440
PR.MSG 34BF    PR.OFF 341A    PR.ON  3414    PRINT  34E4
PRINTR 3400    SAL.SL 3512    RIPETE 3404    RETURN 3406
ROMPRT 300C    ROMTVT 300A    RPLOOP 349B    FINERI 34AB
SELEZ  3205    VAI. 1 3469    VAI. 2 346C    VAI. 3 346F
SPAZIO 347D    SPAZI  3496    TEMP.X 3405    TVSUBS 3100
TVT    3401    TVT.ON 3408    TVTOFF 340E    USR    3402
USR.ON 3420    USROFF 3426    USROUT 300E    VMPAGE 3200
```

C.5 Il dump esadecimale

```
1000            ;       APPENDICE C5: LISTATO ASSEMBLER DEL
1010            ;                DUMP ESADECIMALE
1020            ;
1030            ;
1040            ;
1050            ;              VEDI CAPITOLO 8
1060            ;
1070            ;
1080            ;
1090            ;
1100            ;
1110            ;
1120            ;       COPYRIGHT (C) 1983 DI K .SKIER
1130            ;          LEXINGTON, MASSACHUSETTS
1140            ;
1150            ;
1160            ;
1170            ;
1180            ;
1190            ;
1200            ;
1210            ;
1220            ;
1230            ;
1240            ;
1250            ; ******************************************
1260            ;
1270            ;                 COSTANTI
1280            ;
1290            ; ******************************************
1300            ;
1310            ;
1320            ;
1330            ;
1340            ;
1350                CR      = $0D   RITORNO CARRELLO.
1360            ;
1370                LF      = $0A   'A CAPO'
1380            ;
1390                TEX     = $7F   INDICATORE DI INIZIO
1400            ;                   TESTO.
1410            ;
1420                ETX     = $FF   INDICATORE DI FINE
1430            ;                   TESTO.
1440            ;
1450            ;
1460            ;
1470            ;
1480            ;
1490            ;
1500            ;
1510            ;
1520            ;
1530            ;
1540            ;
1550            ;
1560            ;
1570            ; ******************************************
1580            ;
1590            ;             INDIRIZZI ESTERNI
1600            ;
1610            ; ******************************************
```

```
1620                          ;
1630                          ;
1640                          ;
1650                          ;
1660                          ;
1670                          ;
1680                          ;
1690                          ;
1700                          ;
1710                          ;
1720                               TVSUBS = $3100
1730                                              PAGINA DI PARTENZA
1740                          ;                   DELLE UTILITY VIDEO.
1750
1760                               PULTV  = TVSUBS
1770                               ASCII  = TVSUBS+$B6
1780                          ;
1790                          ;
1800                               VMPAGE = $3200
1810                                              PAGINA DI PARTENZA DEL
1820                          ;                   MONITOR VISIBILE.
1830                          ;
1840                               SELEZ  = VMPAGE+5
1850                               VISMON = VMPAGE+7
1860                               LEGBYT = VMPAGE+$95
1870                               INC.SL = VMPAGE+$10D
1880                          ;
1890                          ;
1900                               PRPAGE = $3400
1910                                              PAGINA DI PARTENZA DELLE
1920                          ;                   UTILITY DI STAMPA.
1930                          ;
1940                               TVT.ON = PRPAGE+8
1950                               TVTOFF = PRPAGE+$0E
1960                               PR.ON  = PRPAGE+$14
1970                               PR.OFF = PRPAGE+$1A
1980                               INVCAR = PRPAGE+$40
1990                               CR.LF  = PRPAGE+$72
2000                               CRLFX  = PRPAGE+$AD
2010                               SPAZIO = PRPAGE+$7D
2020                               SPAZI  = PRPAGE+$96
2030                               PR.BYT = PRPAGE+$83
2040                               INVIA: = PRPAGE+$E4
2050                               SAL.SL = PRPAGE+$112
2060                               REC.SL = PRPAGE+$12B
2070                          ;
2080                          ;
2090                          ;
2100                          ;
2110                          ;
2120                          ;
2130                          ;
2140                          ;
2150                          ;
2160                          ;
2170                          ;
2180                          ;  ******************************************
2190                          ;
2200                          ;                 VARIABILI
2210                          ;
2220                          ;  ******************************************
2230                          ;
2240                          ;
2250                          ;
2260  0000   = 3550                * =   $3550
2270                          ;
```

```
2280
2290                    ;
2300                    ;
2310                    ;
2320  3550  00          CONTAT .BYTE 0        QUESTO BYTE CONTIENE IL
2330                    ;                     NUMERO DI LINEE VISUALIZ-
2340                    ;                     ZATE.
2350  3551  07          MASC   .BYTE 7        PER IL C-64. PER IL VIC
2360                    ;                     USARE .BYTE 3
2370                    ;
2380                    ;
2390                    ;
2400  3552  0000        SA     .WORD 0        PUNTATORE DELL'INDIRIZZO
2410                    ;                     DI PARTENZA.
2420                    ;
2430  3554  FFFF        EA     .WORD $FFFF    PUNTATORE DELL'INDIRIZZO
2440                    ;                     FINALE.
2450                    ;
2460                    ;
2470  3556  00          COLONN .BYTE 0        VARIABILE DI UN BYTE USA-
2480                    ;                     TA COME CONTATORE DA $00
2490                    ;                     A $0F.
2500                    ;
2510                    ;
2520                    ;
2530                    ;
2540                    ;
2550                    ;
2560                    ; ****************************************
2570                    ;
2580                    ;                     TVDUMP
2590                    ;
2600                    ; ****************************************
2610                    ;
2620                    ;
2630                    ;
2640                    ;
2650                    ;
2660  3557  200834      TVDUMP JSR TVT.ON     SELEZIONA IL VIDEO.
2670  355A  A904               LDA #4         SETTA IL CONTATORE CON
2680  355C  8D5035             STA CONTAT     IL NUMERO DI LINEE DA
2690                    ;                     VISUALIZZARE.
2700  355F  AD0532             LDA SELEZ      POSIZIONA SELEZ ALL'INIZIO
2710  3562  29F0               AND #$F0       DI UNA LINEA ESADECIMALE
2720  3564  8D0532             STA SELEZ      (16 BYTE).
2730                    ;
2740                    ;
2750  3567  209B35      DUMPLN JSR PR.ADR     VISUALIZZA L'INDIRIZZO.
2760  356A  207D34             JSR SPAZIO
2770                    ;
2780                    ;
2790  356D  207D34      DMPBYT JSR SPAZIO     FA DUE SPAZI.
2800  3570  209435             JSR DUMPSL     VISUALIZZA IL BYTE.
2810  3573  200D33             JSR INC.SL     SELEZIONA IL SUCCESSIVO.
2820                    ;
2830  3576  AD0532             LDA SELEZ      E' L'INIZIO RIGA SUL
2840  3579  2D5135             AND MASC       VIDEO ?
2850                    ;
2860                    ;
2870                    ;
2880  357C  D0EF               BNE DMPBYT     SE NO, VISUALIZZA IL
2885                    ;                     PROSSIMO BYTE.
2890  357E  207234             JSR CR.LF      SE SI' SI SPOSTA SULLA
2900                    ;                     LINEA SUCCESSIVA.
2910                    ;
2920  3581  AD0532             LDA SELEZ      E' L'INIZIO DI UNA NUOVA
```

```
2930   3584  290F            AND #$0F      LINEA ESADECIMALE?
2940                     ;
2950                     ;
2960   3586  D003            BNE IFFINE
2970   3588  207234          JSR CR.LF     SE SI' SALTA UNA LINEA
2980                     ;                 SULLO SCHERMO.
2990                     ;
3000   358B  CE5035      IFFINE DEC CONTAT ERA L'ULTIMA LINEA ESA?
3010   358E  D0D7            BNE DUMPLN    SE NO, VISUALIZZA LA
3020                     ;                 SUCCESSIVA.
3030                     ;
3040   3590  200E34          JSR TVTOFF    SE SI' DISATTIVA IL
3050                     ;                 VIDEO
3060                     ;
3070   3593  60              RTS           E RITORNA.
3080                     ;
3090                     ;
3100                     ;
3110                     ;
3120                     ;
3130                     ;
3140                     ;
3150                     ;
3160                     ;
3170                     ; *******************************************
3180                     ;
3190                     ;                 DUMPSL
3200                     ;
3210                     ; *******************************************
3220                     ;
3230                     ;
3240                     ;
3250                     ;
3260                     ;
3270   3594  209532      DUMPSL JSR LEGBYT LEGGE IL BYTE SELEZIONATO
3280   3597  208334          JSR PR.BYT    LO VISUALIZZA IN FORMATO
3285                     ;                 ESADECIMALE.
3290   359A  60              RTS           RITORNA.
3300                     ;
3310                     ;
3320                     ;
3330                     ;
3340                     ;
3350                     ;
3360                     ;
3370                     ;
3380                     ;
3390                     ;
3400                     ;
3410                     ; *******************************************
3420                     ;
3430                     ;                 PR.ADR
3440                     ;
3450                     ; *******************************************
3460                     ;
3470                     ;
3480                     ;
3490                     ;
3500                     ;
3510                     ;
3520   359B  AD0632      PR.ADR LDA SELEZ +1 LEGGE IL BYTE ALTO DI
3530   359E  208334          JSR PR.BYT    SELEZ E LO VISUALIZZA.
3540   35A1  AD0532          LDA SELEZ     LEGGE IL BYTE BASSO
3550   35A4  208334          JSR PR.BYT    LO VISUALIZZA
3560   35A7  60              RTS           E RITORNA.
3570                     ;
```

```
3580                        ;
3590                        ;
3600                        ;
3610                        ;
3620                        ;
3630                        ;
3640                        ;
3650                        ;
3660                        ; ****************************************
3670                        ;
3680                        ;                  STAMPA DUMP
3690                        ;
3700                        ; ****************************************
3710                        ;
3720                        ;
3730                        ;
3740                        ;
3750                        ;
3760   35A8  20C335        PRDUMP JSR TITOL      VISUALIZZA IL TITOLO
3770   35AB  20E335               JSR SETADS     SETTA GLI INDIRIZZI DA
3780                        ;                    STAMPARE
3790                        ;
3800                        ;
3810   35AE  209A37               JSR GOTOSA     SETTA SELEZ = SA.
3820   35B1  201434               JSR PR.ON      ATTIVA LA STAMPANTE.
3830                        ;
3840   35B4  20E536               JSR TSTATA     STAMPA LA TESTATA.
3850                        ;
3860                        ;
3870   35B7  203C37        HXLOOP JSR PRLINE     STAMPA UNA LINEA
3880   35BA  10FB                 BPL HXLOOP     FINITO? SE NO, STAMPA LA
3890                        ;                    SUCCESSIVA.
3900                        ;
3910   35BC  207234               JSR CR.LF      SE SI' ... NUOVA RIGA.
3920                        ;
3930   35BF  201A34               JSR PR.OFF     DISATTIVA LA STAMPANTE.
3940                        ;
3950   35C2  60                   RTS            RITORNA.
3960                        ;
3970                        ;
3980                        ;
3990                        ;
4000                        ;
4010                        ;
4020                        ;
4030                        ;
4040                        ;
4050                        ;
4060                        ; ****************************************
4070                        ;
4080                        ;           VISUALIZZA IL TITOLO
4090                        ;
4100                        ; ****************************************
4110                        ;
4120                        ;
4130                        ;
4140                        ;
4150   35C3  200834        TITOL  JSR TVT.ON     ATTIVA IL VIDEO.
4160   35C6  201A34               JSR PR.OFF     DISATTIVA LA STAMPANTE.
4170   35C9  20E434               JSR INVIA:     VISUALIZZA IL MESSAGGIO
4180   35CC  7F                   .BYTE TEX      INIZIO TESTO
4190                        ;
4200   35CD  0D44554D50           .BYTE CR,'DUMP ESADECIMALE',CR,LF,LF    ;
4210         2045534144
4220         4543494D41
4230         4C450D0A0A
```

```
4240    35E1   FF                .BYTE ETX       FINE TESTO.
4250                         ;
4260    35E2   60                RTS             RITORNA.
4270                         ;
4280                         ;
4290                         ;
4300                         ;
4310                         ;
4320                         ;
4330                         ;
4340                         ;
4350                         ;
4360                         ;
4370                         ;  ****************************************
4380                         ;
4390                         ;     PERMETTE DI SPECIFICARE GLI INDIRIZZI
4400                         ;        DI PARTENZA E FINE DI UN BLOCCO
4410                         ;
4420                         ;  ****************************************
4430                         ;
4440                         ;
4450                         ;
4460                         ;
4470                         ;
4480    35E3   200834        SETADS JSR TVT.ON    SELEZIONA IL VIDEO
4490    35E6   20E434               JSR INVIA:    VISUALIZZA LE RICHIESTE
4500    35E9   7F                   .BYTE TEX
4510    35EA   0D0A494E44           .BYTE CR,LF,'INDIRIZZO DI PARTENZA ' ;
4520           4952495A5A
4530           4F20444920
4540           5041525445
4550           4E5A4120
4560    3601   28706F6920           .BYTE '(POI PREMI Q)'            ;
4570           7072656D69
4580           205129
4590    360F   FF                   .BYTE ETX
4600    3610   200732               JSR VISMON    CHIAMA IL M.V. PER SPE-
4610                         ;                    CIFICARE L'INDIRIZZO DEL
4620                         ;                    BLOCCO.
4630                         ;
4640    3613   206136               JSR SAQUI     SETTA L'INDIRIZZO DI PAR-
4650                         ;                    TENZA
4660                         ;
4670                         ;
4680                         ;
4690                         ;
4700                         ;
4710                         ;
4720                         ;
4730                         ;
4740                         ;
4750                         ;
4760                         ;
4770                         ;
4780                         ;
4790    3616   200834        SET.EA JSR TVT.ON    ATTIVA IL VIDEO.
4800    3619   20E434               JSR INVIA:    VISUALIZZA LA RICHIESTA
4810    361C   7F                   .BYTE TEX
4820    361D   0D0A494E44           .BYTE CR,LF,'INDIRIZZO FINALE '   ;
4830           4952495A5A
4840           4F2046494E
4850           414C4520
4860    362F   28706F6920           .BYTE '(POI PREMI Q) ',ETX       ;
4870           7072656D69
4880           20512920FF
4890                         ;
```

```
4900   363E  200732               JSR VISMON     CHIAMA IL MONITOR PER
4910                      ;                      SPECIFICARE L'INDIRIZZO
4920   3641  38                   SEC            TESTA LA VALIDITA' DEI
4930   3642  AD0632               LDA SELEZ +1   DATI.
4940   3645  CD5335               CMP SA+1
4950   3648  9024                 BCC TROBAS
4960   364A  D008                 BNE EAQUI
4970                      ;
4980                      ;
4990                      ;
5000                      ;
5010   364C  AD0532               LDA SELEZ
5020   364F  CD5235               CMP SA
5030   3652  901A                 BCC TROBAS
5040                      ;
5050                      ;
5060                      ;
5070                      ;
5080   3654  AD0632       EAQUI   LDA SELEZ +1   EA = SELEZ .
5090   3657  8D5535               STA EA+1
5100   365A  AD0532               LDA SELEZ
5110   365D  8D5435               STA EA
5120   3660  60                   RTS            RITORNA.
5130                      ;
5140                      ;
5150                      ;
5160                      ;
5170   3661  AD0632       SAQUI   LDA SELEZ +1   SA = SELEZ.
5180   3664  8D5335               STA SA+1
5190   3667  AD0532               LDA SELEZ
5200   366A  8D5235               STA SA
5210   366D  60                   RTS            RITORNA.
5220                      ;
5230   366E  20E434       TROBAS  JSR INVIA:     VISUALIZZA IL MESSAGGIO
5240                      ;                      DI ERRORE
5250                      ;
5260   3671  7F                   .BYTE TEX
5270   3672  0D0A0A0A20           .BYTE CR,LF,LF,LF,'ERRORE. L'INDIRIZZO'
5280         4552524F52           .BYTE 'FINALE E' MINORE DI QUELLO '
5290         452E204C27
5300   3680  494E444952           .BYTE 'DI PARTENZA. CHE E'- '
5310         495A5A4F20
5320         46494E414C
5330         4520452720
5340         4D494E4F52
5350         4520444920
5360         5155454C4C
5370         4F20444920
5380   36A4  5041525445           .BYTE ETX
5390         4E5A412E20
5400         4348452045
5405         27AD20FF
5410   36AF  20B536               JSR PR.SA      VISUALIZZA L'INDIRIZZO
5420                      ;
5430   36B2  4C1636               JMP SET.EA     RICHIEDE L'INDIRIZZO FI-
5440                      ;                      NALE.
5450                      ;
5460                      ;
5470                      ;
5480                      ;
5490                      ;
5500                      ;
5510                      ;
5520                      ;
5530                      ;
5540                      ; ****************************************
```

```
5550                            ;
5560                            ;                   PR.SA
5570                            ;
5580                            ;   ******************************************
5590                            ;
5600                            ;
5610                            ;
5620                            ;
5630    36B5   A924            PR.SA   LDA #'$'        VISUALIZZA IL CARATTERE $
5640    36B7   204034                  JSR INVCAR      PER INDICARE ESADECIMALE.
5650                            ;
5660    36BA   AD5335                  LDA SA+1        BYTE ALTO DELL'INDIRIZZO
5670    36BD   208334                  JSR PR.BYT      DI PARTENZA
5680                            ;
5690    36C0   AD5235                  LDA SA          BYTE BASSO.
5700    36C3   208334                  JSR PR.BYT
5710    36C6   60                      RTS             RITORNA.
5720                            ;
5730                            ;
5740                            ;
5750                            ;
5760                            ;
5770                            ;
5780                            ;
5790                            ;   ******************************************
5800                            ;
5810                            ;                   PR.EA
5820                            ;
5830                            ;   ******************************************
5840                            ;
5850                            ;
5860                            ;
5870                            ;
5880                            ;
5890    36C7   A924            PR.EA   LDA #'$'        VISUALIZZA IL CARATTERE $
5900    36C9   204034                  JSR INVCAR      PER INDICARE ESADECIMALE
5910    36CC   AD5535                  LDA EA+1        BYTE ALTO DELL'INDIRIZZO
5920    36CF   208334                  JSR PR.BYT      FINALE.
5930    36D2   AD5435                  LDA EA          BYTE BASSO.
5940    36D5   208334                  JSR PR.BYT
5950    36D8   60                      RTS             RITORNA.
5960                            ;
5970                            ;
5980                            ;
5990                            ;
6000                            ;
6010                            ;
6020                            ;
6030                            ;
6040                            ;
6050                            ;
6060                            ;   ******************************************
6070                            ;
6080                            ;                   INTERV
6090                            ;
6100                            ;   ******************************************
6110                            ;
6120                            ;
6130                            ;
6140                            ;
6150                            ;
6160    36D9   20B536          INTERV  JSR PR.SA       VISUALIZZA LA PARTENZA.
6170    36DC   A92D                    LDA #'-'        VISUALIZZA UN TRATTINO
6180    36DE   204034                  JSR INVCAR
6190    36E1   20C736                  JSR PR.EA       VISUALIZZA LA FINE
6200    36E4   60                      RTS             RITORNA.
```

```
6210                          ;
6220                          ;
6230                          ;
6240                          ;
6250                          ;
6260                          ;
6270                          ;
6280                          ;
6290                          ;
6300                          ;
6310                          ; ****************************************
6320                          ;
6330                          ;                   LA TESTATA
6340                          ;
6350                          ; ****************************************
6360                          ;
6370                          ;
6380                          ;
6390                          ;
6400                          ;
6410   36E5  20E434          TSTATA JSR INVIA:
6420   36E8  7F                     .BYTE TEX
6430   36E9  0D0A0A4455             .BYTE CR,LF,LF,'DUMP ';
6440         4D5020
6450                          ;
6460   36F4  FF                     .BYTE ETX
6470   36F5  20D936                 JSR INTERV
6480   36F8  207234                 JSR CR.LF
6490   36FB  20E434                 JSR INVIA:
6500   36FE  7F0A0A                 .BYTE TEX,LF,LF
6510   3701  2020202020             .BYTE '       0  1  2  3  4  5  6  7  ';
6520         2020203020
6530         2031202032
6540         2020332020
6550         3420203520
6560         2036202037
6570         2020
6580   3721  3820203920             .BYTE '8  9  A  B  C  D  E  F'          ;
6590         2041202042
6600         2020432020
6610         4420204520
6620         2046
6630   3737  0D0A0AFF               .BYTE CR,LF,LF,ETX
6640   373B  60                     RTS
6650                          ;
6660                          ;
6670                          ;
6680                          ;
6690                          ;
6700                          ;
6710                          ;
6720                          ;
6730                          ;
6740                          ;
6750                          ; ****************************************
6760                          ;
6770                          ;                    PRLINE
6780                          ;
6790                          ; ****************************************
6800                          ;
6810                          ;
6820                          ;
6830                          ;
6840                          ;
6850   373C  207234          PRLINE JSR CR.LF      SALTA UNA RIGA
6860   373F  AD0532                 LDA SELEZ      DETERMINA LA POSIZIONE DI
```

```
6870  3742  48              PHA            PARTENZA.
6880  3743  290F            AND #$0F
6890  3745  8D5635          STA COLONN     ADESSO COLONN CONTIENE IL
6900                    ;                  NUMERO DELLA COLONNA DOVE
6910                    ;                  INIZIARE LA STAMPA.
6920  3748  68              PLA            SETTA SELEZ ALL'INIZIO
6930  3749  29F0            AND #$F0       DELLA LINEA ESADECIMALE.
6940  374B  8D0532          STA SELEZ
6950  374E  209B35          JSR  PR.ADR    STAMPA L'INDIRIZZO .
6960  3751  A203            LDX #3         LASCIA 3 SPAZI.
6970  3753  209634          JSR SPAZI
6980                    ;
6990                    ;
7000  3756  AD5635          LDA COLONN     STAMPA NELLA PRIMA COLON-
7010                    ;                  NA?
7020  3759  F00D            BEQ COL.OK     SE SI' LA POSIZIONE E'
7030                    ;                  GIUSTA
7040                    ;
7050  375B  A203        LOOP    LDX #3     SE NO, LASCIA 3 SPAZI PER
7060  375D  209634          JSR SPAZI      OGNI BYTE DA SALTARE.
7070  3760  200D33          JSR INC.SL
7080  3763  CE5635          DEC COLONN
7090  3766  D0F3            BNE LOOP
7100                    ;
7110  3768  209435      COL.OK JSR DUMPSL  STAMPA IL BYTE SELEZIO-
7120  376B  207D34          JSR SPAZIO     NATO E LASCIA UNO SPAZIO
7130  376E  207D37          JSR NEXTSL     SELEZIONA IL SUCCESSIVO
7140                    ;                  SE NON E' ARRIVATO ALL'IN-
7150  3771  3009            BMI FINE       DIRIZZO DI FINE.
7160                    ;
7170                    ;
7180                    ;
7190  3773  AD0532      NOT.EA LDA SELEZ   FINITA LA LINEA?
7200  3776  290F            AND #$0F       (4 LSB DI SELEZ = 0?)
7210  3778  C900            CMP #0         SE SI' SIAMO A FINE
7220                    ;                  LINEA. SE NO, SELEZIONA
7230  377A  D0EC            BNE COL.OK     IL SUCCESSIVO BYTE
7240  377C  60          FINE    RTS        RITORNA.
7250                    ;
7260                    ;
7270                    ;
7280                    ;
7290                    ;
7300                    ;
7310                    ;
7320                    ;
7330                    ;
7340                    ;
7350                    ;
7360                    ; ****************************************
7370                    ;
7380                    ;                  NEXTSL
7390                    ;
7400                    ; ****************************************
7410                    ;
7420                    ;
7430                    ;
7440                    ;
7450                    ;
7460  377D  38          NEXTSL SEC         SI PREPARA A CONFRONTARE.
7470  377E  AD0632          LDA SELEZ +1   IL BYTE ALTO DI SELEZ E'
7480  3781  CD5535          CMP EA+1       < DEL BYTE ALTO DI EA?
7490  3784  900B            BCC SL.OK      SE SI' SELEZ PUO' ESSERE
7500  3786  D00F            BNE NO.INC     INCREMENTATO. SE NO
7510                    ;                  SELEZ E' NELLA STESSA PA-
7520                    ;                  GINA DI EA.
```

```
7530   3788  38                 SEC           SI PREPARA A COMPARARE IL
7540   3789  AD0532             LDA SELEZ     BYTE BASSO.
7550   378C  CD5435             CMP EA
7560   378F  B006               BCS NO.INC    SE NON E' MINORE NON
7570                     ;                    INCREMENTA.
7580   3791  200D33      SL.OK  JSR INC.SL    POICHE' SELEZ E' MINORE
7590                     ;                    POSSIAMO INCREMENTARLO
7600                     ;
7610   3794  A900               LDA #0        SETTA IL CODICE DI RITOR-
7620   3796  60                 RTS           NO 'PIU' ' E RITORNA.
7630                     ;
7640   3797  A9FF        NO.INC LDA #$FF      SETTA IL CODICE DI RITOR-
7650   3799  60                 RTS           NO 'MENO' E RITORNA.
7660                     ;
7670                     ;
7680                     ;
7690                     ;
7700                     ;
7710                     ;
7720                     ; ******************************************
7730                     ;
7740                     ;                   GOTOSA
7750                     ;
7760                     ; ******************************************
7770                     ;
7780                     ;
7790                     ;
7800                     ;
7810                     ;
7820   379A  AD5235      GOTOSA LDA SA        SETTA SELEZ = SA.
7830   379D  8D0532             STA SELEZ
7840   37A0  AD5335             LDA SA+1
7850   37A3  8D0632             STA SELEZ +1
7860   37A6  60                 RTS           E RITORNA.
```

CROSS REFERENCE

```
ASCII  31B6    PULTV  3100    COL.OK 3768    COLONN 3556
CONTAT 3550    CR     000D    CR.LF  3472    CRLFX  34AD
DMPBYT 356D    DUMPLN 3567    DUMPSL 3594    EA     3554
EAQUI  3654    ETX    00FF    FINE   377C    LEGBYT 3295
GOTOSA 379A    TSTATA 36E5    HXLOOP 35B7    IFFINE 358B
INC.SL 330D    LF     000A    LOOP 375B      MASC   3551
NEXTSL 377D    NO.INC 3797    NOT.EA 3773    REC.SL 352B
PR.ADR 359B    PR.BYT 3483    INVCAR 3440    PR.EA  36C7
PR.OFF 341A    PR.ON  3414    PR.SA  36B5    PRDUMP 35A8
INVIA: 34E4    PRLINE 373C    PRPAGE 3400    SAL.SL 3512
INTERV 36D9    SA     3552    SAQUI  3661    SELEZ  3205
SET.EA 3616    SETADS 35E3    SL.OK  3791    SPAZIO 347D
SPAZI  3496    TEX    007F    TITOL  35C3    TROBAS 366E
TVDUMP 3557    TVSUBS 3100    TVT.ON 3408    TVTOFF 340E
VISMON 3207    VMPAGE 3200
```

C.6 Un disassemblatore
(primo livello e subroutine di utility)

```
1000                    ;       APPENDICE C6: LISTATO ASSEMBLER DI
1010                    ;                     UN DISASSEMBLATORE
1020                    ;
1030                    ;             PRIMO LIVELLO E UTILITY
1040                    ;
1050                    ;
1060                    ;
1070                    ;
1080                    ;                VEDI CAPITOLO 9
1090                    ;
1100                    ;
1110                    ;
1120                    ;
1130                    ;
1140                    ;
1150                    ;          COPYRIGHT (C) 1983 DI K. SKIER
1160                    ;             LEXINGTON, MASSACHUSETTS
1170                    ;
1180                    ;
1190                    ;
1200                    ;
1210                    ;
1220                    ;
1230                    ;
1240                    ;
1250                    ;
1260                    ;
1270                    ;
1280                    ;
1290                    ; *******************************************
1300                    ;
1310                    ;                   COSTANTI
1320                    ;
1330                    ; *******************************************
1340                    ;
1350                    ;
1360                    ;
1370                    ;
1380                    ;
1390                       CR      = $0D   RITORNO CARRELLO.
1400                    ;
1410                       LF      = $0A   'A CAPO'
1420                    ;
1430                    ;
1440                       TEX     = $7F   INDICATORE DI INIZIO
1450                    ;                  TESTO.
1460                    ;
1470                       ETX     = $FF   INDICATORE DI FINE TESTO
1480                    ;
1490                    ;
1500                    ;
1510                    ;
1520                    ;
1530                    ;
1540                    ;
1550                    ;
1560                    ;
1570                    ;
```

```
1580       ; ******************************************
1590       ;
1600       ;          INDIRIZZI ESTERNI
1610       ;
1620       ; ******************************************
1630       ;
1640       ;
1650       ;
1660       ;
1670       ;
1680          PARAMS = $3000
                             PARAMETRI DEL SISTEMA
1690       ;
1700       ;
1710          TVCOL  = PARAMS+3
1720       ;
1730       ;
1740          VMPAGE = $3200
                             PAGINA DI PARTENZA DEL
1750       ;                 MONITOR VISIBILE.
1760       ;
1770
1780          SELEZ  = VMPAGE+5
1790          VISMON = VMPAGE+7
1800          LEGBYT = VMPAGE+$95
1810          INC.SL = VMPAGE+$10D
1820          DEC.SL = VMPAGE+$11A
1830       ;
1840       ;
1850          PRPAGE = $3400
                             PAGINA DI PARTENZA DELLE
1860       ;                 UTILITY DI STAMPA
1870       ;
1880
1890          TVT.ON = PRPAGE+8
1900          TVTOFF = PRPAGE+$0E
1910          PR.ON  = PRPAGE+$14
1920          PR.OFF = PRPAGE+$1A
1930          PR.CAR = PRPAGE+$40
1940          CR.LF  = PRPAGE+$72
1950          SPAZIO = PRPAGE+$7D
1960          SPAZI  = PRPAGE+$96
1970          PR.BYT = PRPAGE+$83
1980          INVIA: = PRPAGE+$E4
1990          SAL.SL = PRPAGE+$112
2000          REC.SL = PRPAGE+$12B
2010       ;
2020       ;
2030          HEX.PG = $3500
                             INDIRIZZO DELLA PAGINA DI
2040       ;                 PARTENZA DEL DUMP ESADE-
2050       ;                 CIMALE.
2060       ;
2070          SA     = HEX.PG+$52
2080          EA     = SA+2
2090          DUMPSL = HEX.PG+$94
2100          PR.ADR = HEX.PG+$9B
2110          INTERV = HEX.PG+$1D9
2120          SETADS = HEX.PG+$E3
2130          NEXTSL = HEX.PG+$27D
2140          GOTOSA = HEX.PG+$29A
2150       ;
2160       ;
2170       ;
2180       ;
2190       ;
2200       ;    TABELLE DEL DISASSEMBLER
2210       ;
2220       ;
2230       ;
```

```
2240                             DSPAGE = $3900
2250                        ;                       PAGINA DI PARTENZA DEL
2260                        ;                       DISASSEMBLER.
2270                             SUBS   = DSPAGE+$226
2280                             MNOMI  = DSPAGE+$250
2290                             MCODES = DSPAGE+$300
2300                             MODI   = DSPAGE+$400
2310                        ;
2320                        ;
2330                        ;
2340                        ;
2350                        ; ****************************************
2360                        ;
2370                        ;               VARIABILI
2380                        ;
2390                        ; ****************************************
2400                        ;
2410                        ;
2420                        ;
2430    0000  = 3900             * =  DSPAGE
2440                        ;
2450                        ;
2460                        ;
2470                        ;
2480                        ;
2490    3900  05            DISLNX .BYTE 5        STABILISCE IL NUMERO DI
2500                        ;                     LINEE DA DISASSEMBLARE
2510                        ;
2520    3901  00            LINUM  .BYTE 0        VARIABILE CHE TIENE CONTO
2530                        ;                     DELLE LINEE DISASSEMBLATE.
2540    3902  00            LETTER .BYTE 0        CONTATORE DELLE LETTERE
2550                        ;                     STAMPATE.
2560                        ;
2570    3903  00            TEMP.X .BYTE 0
2580                        ;
2590    3904  0000          SUBPTR .WORD 0
2600                        ;
2610                        ;
2620    3906  00            OPBYT  .BYTE 0        VARIABILE USATA DA FINITO.
2630                        ;
2640    3907  00            OPCAR  .BYTE 0        VARIABILE USATA DA FINITO.
2650                        ;
2660    3908  0B            INDCOL .BYTE 11       COLONNA DI PARTENZA DEL
2670                        ;                     CAMPO INDIRIZZI.
2680                        ;
2690                        ;
2700                        ;
2710                        ;
2720                        ;
2730                        ;
2740                        ;
2750                        ;
2760                        ; ****************************************
2770                        ;
2780                        ;          DISASSEMBLER SU VIDEO
2790                        ;
2800                        ; ****************************************
2810                        ;
2820                        ;
2830                        ;
2840                        ;
2850                        ;
2860    3909  200834        TV.DIS JSR TVT.ON     SELEZIONA IL VIDEO.
2870    390C  AD0039               LDA DISLNX     INIZIALIZZA IL CONTATORE
2880    390F  8D0139               STA LINUM      COL NUMERO DI LINEE DA
2890                        ;                     DISASSEMBLARE.
```

```
2900   3912  A9FF                     LDA #$FF       SETTA L'INDIRIZZO DI FINE
2910   3914  8D5435                   STA EA         A $FFFF PER POTER SEMPRE
2920   3917  8D5535                   STA EA+1       INCREMENTARE IL PUNTATORE
2930   391A  207234                   JSR CR.LF
2940                          ;
2950   391D  207D39           TVLOOP  JSR DSLINE     DISASSEMBLA UNA LINEA.
2960   3920  CE0139                   DEC LINUM      E' L'ULTIMA?
2970   3923  D0F8                     BNE TVLOOP     SE NO, PROSEGUE.
2980   3925  60                       RTS            SE SI' RITORNA.
2990                          ;
3000                          ;
3010                          ;
3020                          ;
3030                          ;
3040                          ;
3050                          ;
3060                          ;
3070                          ;
3080                          ;  ****************************************
3090                          ;
3100                          ;          DISASSEMBLER SU STAMPANTE
3110                          ;
3120                          ;  ****************************************
3130                          ;
3140                          ;
3150                          ;
3160                          ;
3170                          ;
3180                          ;
3190   3926  201A34           PR.DIS  JSR PR.OFF     DISATTIVA LA STAMPANTE
3200   3929  200834                   JSR TVT.ON     ATTIVA IL VIDEO.
3210   392C  20E434                   JSR INVIA:     VISUALIZZA IL TITOLO.
3220   392F  7F0D0A                   .BYTE TEX,CR,LF
3230   3932  4449534153               .BYTE 'DISASSEMBLER SU STAMPANTE' ;
3240         53454D424C
3250         4552205355
3260         205354414D
3270         50414E5445
3280
3290   394D  0D0AFF                   .BYTE CR,LF,ETX
3300                          ;
3310   3950  20E335                   JSR SETADS     LEGGE GLI INDIRIZZI DI
3320                          ;                      PARTENZA E FINE SPECIFI-
3330                          ;                      CATI DALL'UTENTE.
3340   3953  201434                   JSR PR.ON      ATTIVA LA STAMPANTE.
3350   3956  20E434                   JSR INVIA:
3360   3959  7F0D0A                   .BYTE TEX,CR,LF
3370   395C  4449534153               .BYTE 'DISASSEMBLATO ';
3380         53454D424C
3390         41544F20
3400   396A  FF                       .BYTE ETX
3410   396B  20D936                   JSR INTERV     STAMPA GLI INDIRIZZI DI
3420                          ;                      INIZIO E FINE BLOCCO.
3430   396E  209A37                   JSR GOTOSA     SETTA SELEZ ALL'INIZIO
3440                          ;                      DEL BLOCCO.
3450                          ;
3460   3971  207234                   JSR CR.LF      AVANZA DI UNA RIGA.
3470   3974  207D39           PRLOOP  JSR DSLINE     DISASSEMBLA UNA LINEA.
3480   3977  10FB                     BPL PRLOOP     SE NON ERA L'ULTIMA
3490                          ;                      DISASSEMBLA LA SUCCESSIVA
3500                          ;
3510                          ;
3520   3979  201A34                   JSR PR.OFF     DISATTIVA LA STAMPANTE.
3530                          ;
3540   397C  60                       RTS            RITORNA.
3550                          ;
```

```
3560                    ;
3570                    ;
3580                    ;
3590                    ;
3600                    ;
3610                    ;
3620                    ;
3630                    ;
3640                    ;
3650                    ; ******************************************
3660                    ;
3670                    ;              DISASSEMBLA UNA LINEA
3680                    ;
3690                    ; ******************************************
3700                    ;
3710                    ;
3720                    ;
3730                    ;
3740                    ;
3750   397D  209532     DSLINE JSR LEGBYT     LEGGE IL BYTE SELEZIONATO.
3760   3980  48                PHA            LO SALVA NELLO STACK.
3770   3981  209239            JSR MNEMON     STABILISCE LO MNEMONICO
3780                    ;                     CORRISPONDENTE.
3790   3984  207D34            JSR SPAZIO     LASCIA UNO SPAZIO.
3800   3987  68                PLA            RIMETTE L'OPCODE IN A.
3810   3988  20AF39            JSR OPERND     STABILISCE L'OPERANDO
3820                    ;                     RICHIESTO.
3830   398B  20013A            JSR FINITO     TERMINA LA LINEA VISUALIZ-
3840                    ;                     ZANDO I CAMPI DA 3 A 6.
3850                    ;
3860                    ;
3870                    ;
3880   398E  207D37            JSR NEXTSL     SELEZIONA IL PROSSIMO
3890                    ;                     BYTE.
3900   3991  60                RTS            RITORNA.
3910                    ;
3920                    ;
3930                    ;
3940                    ;
3950                    ;
3960                    ;
3970                    ;
3980                    ;
3990                    ;
4000                    ;
4010                    ; ******************************************
4020                    ;
4030                    ;              STAMPA LO MNEMONICO
4040                    ;
4050                    ; ******************************************
4060                    ;
4070                    ;
4080                    ;
4090                    ;
4100                    ;
4110   3992  A203       MNEMON LDX #3         CI SONO 3 LETTERE IN UNO
4120   3994  8E0239            STX LETTER     MNEMONICO.
4130   3997  AA                TAX            SI PREPARA AD USARE L'OP-
4140                    ;                     CODE COME INDICE.
4150                    ;
4160   3998  BD003C            LDA MCODES,X   CERCA IL CODICE MNEMONICO
4170                    ;                     DELL'OPCODE (MCODES E' LA
4180                    ;                     TABELLA DEI CODICI).
4190                    ;
4200   399B  AA                TAX            SI PREPARA AD USARE IL CO-
4210                    ;                     DICE TROVATO COME INDICE.
```

```
4220   399C  BD503B     MNLOOP LDA MNOMI ,X   LEGGE IL NOME DELLO MNE-
4230                    ;                     MONICO CORRISPONDENTE.
4240                    ;
4250                    ;
4260   399F  8E0339            STX TEMP.X
4270                    ;
4280   39A2  204034            JSR PR.CAR     STAMPA IL CARATTERE.
4290   39A5  AE0339            LDX TEMP.X
4300   39A8  E8                INX            LEGGE LA LETTERA SUCCESSIVA.
4310   39A9  CE0239            DEC LETTER     SE NON E' LA TERZA LETTERA
4320   39AC  D0EE              BNE MNLOOP     DEL NOME RIENTRA NEL LOOP,
4330   39AE  60                RTS            SE SI' RITORNA.
4340                    ;
4350                    ;
4360                    ;
4370                    ;
4380                    ;
4390                    ;
4400                    ;
4410                    ;
4420                    ;
4430                    ;
4440                    ; ******************************************
4450                    ;
4460                    ;             STAMPA L'OPERANDO
4470                    ;
4480                    ; ******************************************
4490                    ;
4500                    ;
4510                    ;
4520                    ;
4530                    ;
4540   39AF  AA         OPERND TAX            CERCA IL MODO DI INDI-
4550   39B0  BD003D            LDA MODI ,X    RIZZAMENTO DELL'OPCODE.
4560                    ;
4570   39B3  AA                TAX            ADESSO X INDICA IL MODO
4580                    ;
4590                    ;
4600   39B4  20B839            JSR MODO.X     CHIAMA LA SUBROUTINE DEI
4610   39B7  60                RTS            MODI E RITORNA.
4620                    ;
4630                    ;
4640                    ;
4650                    ;
4660                    ;
4670                    ;
4680                    ;
4690                    ;
4700                    ;
4710                    ;
4720                    ; ******************************************
4730                    ;
4740                    ;                 MODO "X"
4750                    ;
4760                    ; ******************************************
4770                    ;
4780                    ;
4790                    ;
4800                    ;
4810                    ;
4820                    ;
4830   39B8  BD263B     MODO.X LDA SUBS,X     LEGGE IL BYTE BASSO DEL-
4840   39BB  8D0439            STA SUBPTR     L'X-ESIMO PUNTATORE.
4850                    ;
4860   39BE  E8                INX            INCREMENTA L'INDICE PER
4870   39BF  BD263B            LDA SUBS,X     LEGGERE IL BYTE SUCCESSIVO
```

```
4880   39C2  8D0539             STA SUBPTR+1
4890   39C5  6C0439             JMP (SUBPTR)    TRASFERISCE IL CONTROLLO
4900                     ;                      ALLA ROUTINE RICHIESTA
4910                     ;                      CHE NON RITORNERA' QUI MA
4920                     ;                      AL PROGRAMMA CHE HA
4930                     ;                      RICHIAMATO MODO.X
4940                     ;
4950                     ;
4960                     ;
4970                     ;
4980                     ;
4990                     ;
5000                     ;
5010                     ;
5020                     ;
5030                     ;
5040                     ; ****************************************
5050                     ;
5060                     ;        UTILITY DEL DISASSEMBLATORE
5070                     ;
5080                     ; ****************************************
5090                     ;
5100                     ;
5110                     ;
5120                     ;
5130                     ;
5140                     ;                      ONEBYT
5150                     ;
5160                     ;
5170                     ;
5180   39C8  200D33      ONEBYT JSR INC.SL      AVANZA AL PROSSIMO OPCODE
5190                     ;
5200   39CB  209435             JSR DUMPSL      LO STAMPA IN ESADECIMALE
5210   39CE  60                 RTS             RITORNA.
5220                     ;
5230                     ;
5240                     ;
5250                     ;
5260                     ;
5270                     ;                      TWOBYT
5280                     ;
5290                     ;
5300                     ;
5310   39CF  200D33      TWOBYT JSR INC.SL      AVANZA AL PRIMO BYTE
5320                     ;                      DELL'OPERANDO.
5330   39D2  209532             JSR LEGBYT      CARICA IL BYTE IN A.
5340   39D5  48                 PHA             LO SALVA.
5350   39D6  200D33             JSR INC.SL      LEGGE IL SECONDO BYTE
5360                     ;
5370   39D9  209435             JSR DUMPSL      LO STAMPA IN ESADECIMALE
5380   39DC  68                 PLA             RICHIAMA IL PRIMO BYTE
5390   39DD  208334             JSR PR.BYT      LO STAMPA IN ESADECIMALE
5400   39E0  60                 RTS             RITORNA.
5410                     ;
5420                     ;
5430                     ;
5440                     ;
5450                     ;
5460                     ;                      RPAREN-LPAREN
5470                     ;
5480                     ;
5490                     ;
5500   39E1  A928        LPAREN LDA #$28        STAMPA UNA PARENTESI
5510   39E3  D002               BNE MANDA       APERTA.
5520                     ;
5530                     ;
```

```
5540   39E5  A929       RPAREN LDA #$29       STAMPA UNA PARENTESI
5550                     ;                    CHIUSA.
5560   39E7  204034     MANDA  JSR PR.CAR
5570   39EA  60                RTS            RITORNA.
5580                     ;
5590                     ;
5600                     ;
5610                     ;
5620                     ;                    XINDEX
5630                     ;
5640                     ;
5650                     ;
5660                     ;
5670   39EB  A92C       XINDEX LDA #','
5680   39ED  204034            JSR PR.CAR     STAMPA UNA VIRGOLA
5690   39F0  A958              LDA #'X'
5700   39F2  204034            JSR PR.CAR     SEGUITA DA 'X'.
5710   39F5  60                RTS
5720                     ;
5730                     ;
5740                     ;
5750                     ;
5760                     ;                    YINDEX
5770                     ;
5780                     ;
5790                     ;
5800                     ;
5810   39F6  A92C       YINDEX LDA #','
5820   39F8  204034            JSR PR.CAR     STAMPA UNA VIRGOLA
5830   39FB  A959              LDA #'Y'
5840   39FD  204034            JSR PR.CAR     SEGUITA DA 'Y'.
5850   3A00  60                RTS
5860                     ;
5870                     ;
5880                     ;
5890                     ;
5900                     ;
5910                     ;
5920                     ;
5930                     ;
5940                     ;
5950                     ;
5960                     ;  *****************************************
5970                     ;
5980                     ;                    FINITO
5990                     ;
6000                     ;  *****************************************
6010                     ;
6020                     ;
6030                     ;
6040                     ;        NOTA:       OGNI ROUTINE DI INDI-
6050                     ;                    RIZZAMENTO DEVE FINIRE
6060                     ;                    PONENDO X UGUALE AL NU-
6070                     ;                    MERO DI BYTE DELL'OPE-
6080                     ;                    RANDO, E L'ACCUMULATO-
6090                     ;                    RE UGUALE AL NUMERO DEI
6100                     ;                    CARATTERI DELL'OPERANDO.
6110                     ;
6120                     ;
6130   3A01  8D0739     FINITO STA OPCAR      SALVA LA LUNGHEZZA DEL-
6140   3A04  8E0639            STX OPBYT      L'OPERANDO IN CARATTERI
6150                     ;                    E IN BYTE
6160                     ;
6170                     ;
6180   3A07  CA                DEX            SE NECESSARIO DECREMENTA
6190                     ;                    SELEZ PERCHE' PUNTI L'OP-
```

```
6200   3A08  3006           BMI SEL.OK      CODE.
6210   3A0A  201A33  LOOP.1 JSR DEC.SL
6220   3A0D  CA             DEX
6230   3A0E  10FA           BPL LOOP.1
6240                 ;
6250
6260                 ;
6270   3A10  08      SEL.OK PHP             SALVA IL FLAG DECIMALE E
6280   3A11  D8             CLD             SI PREPARA PER LA SOMMA.
6290                 ;
6300   3A12  38             SEC             LASCIA GLI SPAZI NECES-
6310   3A13  AD0839         LDA INDCOL      SARI AL CAMPO INDIRIZZI
6320   3A16  E904           SBC #4          L'OPERANDO PARTE ALLA
6330                 ;                      COLONNA 4...
6340   3A18  ED0739         SBC OPCAR       E INCLUDE OPCAR CARATTERI
6350                 ;
6360   3A1B  28             PLP             RESETTA IL FLAG DECIMALE
6370   3A1C  AA             TAX
6380   3A1D  209634         JSR SPAZI       LASCIA GLI SPAZI NECES-
6390                 ;                      SARI A RAGGIUNGERE LA
6400                 ;                      COLONNA DEGLI INDIRIZZI
6410   3A20  209B35         JSR PR.ADR      STAMPA L'INDIRIZZO DEL-
6420   3A23  207D34         JSR SPAZIO      L'OPCODE.
6430                 ;
6440   3A26  209435  LOOP.2 JSR DUMPSL      STAMPA IL BYTE.
6450
6460   3A29  AD0330         LDA TVCOL       SE LO SCHERMO HA MENO DI
6470   3A2C  38             SEC             24 COLONNE NON LASCIA
6480   3A2D  C918           CMP #24         UN ALTRO SPAZIO
6490   3A2F  9003           BCC DUMPED
6500                 ;
6510                 ;                      SE LO SCHERMO HA PIU' DI
6520                 ;                      24 COLONNE LASCIA UNO
6530   3A31  207D34         JSR SPAZIO      SPAZIO DOPO AVER STAMPATO
6540                 ;                      IL BYTE.
6550   3A34          DUMPED
6560   3A34  200D33         JSR INC.SL      SELEZIONA IL PROSSIMO BYTE
6570   3A37  CE0639         DEC OPBYT       E' L'ULTIMO BYTE DELL'IS-
6580                 ;                      TRUZIONE?
6590   3A3A  10EA           BPL LOOP.2      SE NO, TORNA AL LOOP
6600   3A3C  201A33         JSR DEC.SL      RIPORTA SELEZ ALL'ULTIMO
6610                 ;                      BYTE DELL'OPERANDO.
6620                 ;
6630                 ;
6640                 ;
6650                 ;
6660   3A3F  207234  FINEND JSR CR.LF       SE SI' VA A CAPO.
6670                 ;
6680   3A42  60             RTS             RITORNA.
6690                 ;
```

CROSS REFERENCE

```
INDCOL 3908    CR     000D    CR.LF  3472    DEC.SL 331A
DISLNX 3900    DSLINE 397D    DSPAGE 3900    DUMPED 3A34
DUMPSL 3594    EA     3554    ETX    00FF    FINEND 3A3F
FINITO 3A01    LEGBYT 3295    GOTOSA 379A    HEX.PG 3500
INC.SL 330D    LETTER 3902    LF     000A    LINUM  3901
LOOP.1 3A0A    LOOP.2 3A26    LPAREN 39E1    MCODES 3C00
MNOMI  3B50    MNEMON 3992    MNLOOP 399C    MODO.X 39B8
MODI   3D00    NEXTSL 377D    ONEBYT 39C8    OPBYT  3906
```

```
OPCAR  3907    OPERND 39AF    PARAMS 3000    REC.SL 352B
PR.ADR 359B    PR.BYT 3483    PR.CAR 3440    PR.DIS 3926
PR.OFF 341A    PR.ON  3414    INVIA: 34E4    PRLOOP 3974
PRPAGE 3400    SAL.SL 3512    INTERV 36D9    RPAREN 39E5
SA     3552    SEL.OK 3A10    SELEZ  3205    MANDA  39E7
SETADS 35E3    SPAZIO 347D    SPAZI  3496    SUBPTR 3904
SUBS   3B26    TEMP.X 3903    TEX    007F    TV.DIS 3909
TVCOL  3003    TVLOOP 391D    TVT.ON 3408    TVTOFF 340E
TWOBYT 39CF    VISMON 3207    VMPAGE 3200    XINDEX 39EB
YINDEX 39F6
```

C.7 Un disassemblatore (subroutine per il modo di indirizzamento)

```
1000                ;        APPENDICE C7: LISTATO ASSEMBLER DI
1010                ;                      UN DISASSEMBLATORE
1020                ;
1030                ;            SUBROUTINE DI INDIRIZZAMENTO
1040                ;
1050                ;
1060                ;
1070                ;
1080                ;
1090                ;
1100                ;
1110                ;                   VEDI CAPITOLO 9
1120                ;
1130                ;
1140                ;
1150                ;
1160                ;
1170                ;
1180                ;            COPYRIGHT (C) 1983 DI K. SKIER
1190                ;              LEXINGTON, MASSACHUSETTS
1200                ;
1210                ;
1220                ;
1230                ;
1240                ;
1250                ;
1260                ;
1270                ;
1280                ;
1290                ;
1300                ;
1310                ;
1320                ;
1330                ;
1340                ;
1350                ;
1360                ; ******************************************
1370                ;
1380                ;                   COSTANTI
1390                ;
1400                ; ******************************************
1410                ;
1420                ;
1430                ;
1440                ;
1450                ;
1460                      CR      = $0D   RITORNO CARRELLO
1470                ;
1480                      LF      = $0A   'A CAPO'
1490                ;
1500                ;
1510                      TEX     = $7F   INDICATORE DI INIZIO
1520                ;                     TESTO
1530                ;
1540                      ETX     = $FF   INDICATORE DI FINE
1550                ;                     TESTO
1560                ;
1570                ;
1580                ;
1590                ;
```

```
1600                ;
1610                ;
1620                ;
1630                ;
1640                ;
1650                ;
1660                ;
1670                ;
1680                ;
1690                ; ******************************************
1700                ;
1710                ;          INDIRIZZI ESTERNI
1720                ;
1730                ; ******************************************
1740                ;
1750                ;
1760                ;
1770                ;
1780                ;
1790                ;
1800                ;
1810                ;
1820                ;
1830                ;
1840                ;
1850                        VMPAGE = $3200
1860                ;                       PAGINA DI PARTENZA DEL
1870                ;                       MONITOR VISIBILE.
1880                ;
1890                        SELEZ  = VMPAGE+5
1900                        VISMON = VMPAGE+7
1910                        LEGBYT = VMPAGE+$95
1920                        INC.SL = VMPAGE+$10D
1930                        DEC.SL = VMPAGE+$11A
1940                ;
1950                ;
1960                        PRPAGE = $3400
1970                ;                       PAGINA DI PARTENZA DELLE
1980                ;                       UTILITY DI STAMPA.
1990                ;
2000                        PR.CAR = PRPAGE+$40
2010                        CR.LF  = PRPAGE+$72
2020                        SPAZIO = PRPAGE+$7D
2030                        SPAZI  = PRPAGE+$96
2040                        PR.BYT = PRPAGE+$83
2050                        SAL.SL = PRPAGE+$112
2060                        REC.SL = PRPAGE+$12B
2070                ;
2080                ;
2090                        HEX.PG = $3500
2100                ;                       INDIRIZZO DELLA PAGINA
2110                ;                       DI PARTENZA DEL DUMP
2120                ;                       ESADECIMALE.
2130                        PR.ADR = HEX.PG+$9B
2140                        NEXTSL = HEX.PG+$27D
2150                ;
2160                ;
2170                        DSPAGE = $3900
2180                ;                       INDIRIZZO DI PARTENZA DEL
2190                ;                       DISASSEMBLER.
2200                        ONEBYT = DSPAGE+$C8
2210                        TWOBYT = DSPAGE+$CF
2220                        LPAREN = DSPAGE+$E1
2230                        RPAREN = DSPAGE+$E5
2240                        XINDEX = DSPAGE+$EB
2250                        YINDEX = DSPAGE+$F6
```

```
2260                    ;
2270                    ;
2280                    ;
2290                    ;
2300                    ;
2310                    ;
2320                    ;
2330  0000   = 3A50           * =  DSPAGE+$150
2340                    ;
2350                    ;
2360                    ;
2370                    ;
2380                    ;
2390                    ;
2400                    ;
2410                    ;
2420                    ;
2430                    ;
2440                    ;
2450                    ; ******************************************
2460                    ;
2470                    ;        SUBROUTINE DI INDIRIZZAMENTO
2480                    ;
2490                    ; ******************************************
2500                    ;
2510                    ;
2520                    ;
2530                    ;
2540                    ;
2550                    ;
2560                    ;                      MODO ASSOLUTO
2570                    ;
2580                    ;
2590                    ;
2600  3A50   20CF39     ABSLUT JSR TWOBYT
2610  3A53   A202              LDX #2        X CONTIENE IL NUMERO DI
                                             BYTE DELL'OPERANDO.
2615                    ;
2620  3A55   A904              LDA #4        A CONTIENE IL NUMERO DI
                                             CARATTERI.
2625                    ,
2630  3A57   60                RTS           RITORNA.
2640                    ;
2650                    ;
2660                    ;
2670                    ;
2680                    ;
2690                    ;              MODO ASSOLUTO,X
2700                    ;
2710                    ;
2720                    ;
2730  3A58   20503A     ABS.X  JSR ABSLUT    STAMPA UNA VIRGOLA SE-
2740  3A5B   20EB39            JSR XINDEX    GUITA DA X
2750  3A5E   A202              LDX #2        L'OPERANDO HA 2 BYTE
2760  3A60   A906              LDA #6        E 6 CARATTERI.
2770  3A62   60                RTS           RITORNA.
2780                    ;
2790                    ;
2800                    ;
2810                    ;
2820                    ;
2830                    ;              MODO ASSOLUTO,Y
2840                    ;
2850                    ;
2860                    ;
2870  3A63   20503A     ABS.Y  JSR ABSLUT
2880  3A66   20F639            JSR YINDEX
2890  3A69   A202              LDX #2
```

```
2900  3A6B  A906            LDA #6
2910  3A6D  60              RTS
2920                 ;
2930                 ;
2940                 ;
2950                 ;
2960                 ;
2970                 ;          MODO ACCUMULATORE
2980                 ;
2990                 ;
3000  3A6E  A941     ACC    LDA #'A'        STAMPA IL CARATTERE 'A'
3010  3A70  204034          JSR PR.CAR
3020  3A73  A200            LDX #0          L'OPERANDO HA 0 BYTE
3030  3A75  A901            LDA #1          E UN CARATTERE.
3040  3A77  60              RTS             RITORNA.
3050                 ;
3060                 ;
3070                 ;
3080                 ;
3090                 ;
3100                 ;          MODO IMPLICITO
3110                 ;
3120                 ;
3130                 ;
3140  3A78  A200     IMPLID LDX #0          L'OPERANDO HA 0 BYTE
3150  3A7A  A900            LDA #0          E 0 CARATTERI
3160  3A7C  60              RTS             RITORNA.
3170                 ;
3180                 ;
3190                 ;
3200                 ;
3210                 ;
3220                 ;          MODO IMMEDIATO
3230                 ;
3240                 ;
3250                 ;
3260  3A7D  A923     IMMEDT LDA #'#'        STAMPA IL SIMBOLO '#'.
3270  3A7F  204034          JSR PR.CAR
3280                 ;
3290  3A82  A924            LDA #'$'        STAMPA IL SIMBOLO '$'.
3300  3A84  204034          JSR PR.CAR
3310  3A87  20C839          JSR ONEBYT      STAMPA L'OPERANDO DI UN
3320                 ;                      BYTE IN FORMATO ESADECI-
3330  3A8A  A201            LDX #1          MALE
3340  3A8C  A904            LDA #4
3350  3A8E  60              RTS             RITORNA.
3360                 ;
3370                 ;
3380                 ;
3390                 ;
3400                 ;
3410                 ;          MODO INDIRETTO
3420                 ;
3430                 ;
3440                 ;
3450  3A8F  20E139   INDRCT JSR LPAREN      STAMPA IL CARATTERE '('.
3460  3A92  20503A          JSR ABSLUT
3470  3A95  20E539          JSR RPAREN      STAMPA IL CARATTERE ')'.
3480  3A98  A906            LDA #6          NUMERO DI CARATTERI
3490                 ;                      DELL'OPERANDO.
3500  3A9A  A202            LDX #2          NUMERO DI BYTE
3510                 ;                      DELL'OPERANDO.
3520  3A9C  60              RTS             RITORNA.
3530                 ;
3540                 ;
3550                 ;
```

```
3560                         ;
3570                         ;
3580                         ;          MODO INDIRETTO, X
3590                         ;
3600                         ;
3610                         ;
3620  3A9D   20E139          IND.X   JSR LPAREN     STAMPA IL CARATTERE '('
3630  3AA0   20F33A                  JSR ZERO.X     STAMPA L'INDIRIZZO DI PA-
3640                         ;                      GINA 0, UNA VIRGOLA E 'X'
3650  3AA3   20E539                  JSR RPAREN
3660  3AA6   A201                    LDX #1         STAMPA IL CARATTERE ')'.
3670  3AA8   A906                    LDA #6
3680  3AAA   60                      RTS            RITORNA.
3690                         ;
3700                         ;
3710                         ;
3720                         ;
3730                         ;
3740                         ;       MODO INDIRETTO,Y
3750                         ;
3760                         ;
3770                         ;
3780  3AAB   20E139          IND.Y   JSR LPAREN     STAMPA IL CARATTERE '('
3790  3AAE   20EB3A                  JSR ZEROPG     L'INDIRIZZO DI PAGINA ZE-
3800  3AB1   20E539                  JSR RPAREN     RO E IL CARATTERE ')'.
3810  3AB4   20F639                  JSR  YINDEX    STAMPA ',' SEGUITA DA 'Y'
3820  3AB7   A201                    LDX #1
3830  3AB9   A906                    LDA #6
3840  3ABB   60                      RTS            RITORNA.
3850                         ;
3860                         ;
3870                         ;
3880                         ;
3890                         ;
3900                         ;          MODO RELATIVO
3910                         ;
3920                         ;
3930                         ;
3940  3ABC   200D33          RELATV  JSR INC.SL     SELEZIONA IL BYTE
3950  3ABF   201235                  JSR SAL.SL     SALVA SELEZ NELLO STACK
3960  3AC2   209532                  JSR LEGBYT     LEGGE IL BYTE DELL'OPE-
3970  3AC5   48                      PHA            RANDO E LO SALVA.
3980  3AC6   200D33                  JSR INC.SL     INCREMENTA SELEZ IN MODO
3990                         ;                      CHE PUNTI AL BYTE IMME-
4000                         ;                      DIATAMENTE SUCCESSIVO
4010                         ;                      (ALL'ISTRUZIONE DI SALTO
4015                         ;                      RELATIVO).
4020  3AC9   68                      PLA            RECUPERA IL BYTE OPERANDO
4025                         ;                      DALLO STACK.
4030  3ACA   C900                    CMP #0         E' PIU' O MENO?
4040  3ACC   1003                    BPL AVANTI     SE E'PIU' SIGNIFICA SALTO
4050                         ;                      IN AVANTI.
4060                         ;
4070                         ;
4080                         ;
4090  3ACE   CE0632                  DEC SELEZ +1   UN SALTO ALL'INDIETRO E'
4100                         ;                      UGUALE A UN SALTO IN A-
4110                         ;                      VANTI ESEGUITO DA UNA
4120                         ;                      LOCAZIONE 256 BYTE PIU'
4130                         ;                      BASSA.
4140  3AD1   08              AVANTI  PHP            SALVA IL FLAG DECIMALE.
4150  3AD2   D8                      CLD
4160                         ;
4170  3AD3   18                      CLC            SI PREPARA ALL'ADDIZIONE
4180  3AD4   6D0532                  ADC SELEZ      SOMMA IL BYTE OPERANDO
4190  3AD7   9003                    BCC RELEND     ALL'INDIRIZZO CHE ESEGUE
```

```
4200   3AD9  EE0632          INC SELEZ +1     L'ISTRUZIONE DI SALTO.
4210   3ADC  8D0532   RELEND STA SELEZ        ADESSO SELEZ PUNTA AL-
4220                  ;                       L'INDIRIZZO SPECIFICATO
4230                  ;                       DALL'ISTRUZIONE DI SALTO
4235                  ;                       RELATIVO.
4240   3ADF  28              PLP              RESETTA IL FLAG DECIMALE
4250                  ;
4260   3AE0  209B35          JSR PR.ADR       STAMPA L'INDIRIZZO SPECI-
4270                  ;                       FICATO DALL'ISTRUZIONE.
4280   3AE3  202B35          JSR REC.SL       RESETTA IL PUNTATORE
4290                  ;                       SELEZ
4300   3AE6  A201            LDX #1
4310   3AE8  A904            LDA #4
4320   3AEA  60              RTS              RITORNA.
4330                  ;
4340                  ;
4350                  ;
4360                  ;
4370                  ;           MODO PAGINA ZERO
4380                  ;
4390                  ;
4400                  ;
4410                  ;
4420   3AEB  20C839   ZEROPG JSR ONEBYT       STAMPA L'OPERANDO.
4430   3AEE  A201            LDX #1
4440   3AF0  A902            LDA #2
4450   3AF2  60              RTS              RITORNA.
4460                  ;
4470                  ;
4480                  ;
4490                  ;
4500                  ;
4510                  ;           MODO PAGINA ZERO.X
4520                  ;
4530                  ;
4540                  ;
4550   3AF3  20EB3A   ZERO.X JSR ZEROPG       STAMPA L'INDIRIZZO DI PA-
4560   3AF6  20EB39          JSR XINDEX       GINA ZERO ',' E 'X'
4570   3AF9  A201            LDX #1
4580   3AFB  A904            LDA #4
4590   3AFD  60              RTS              RITORNA.
4600                  ;
4610                  ;
4620                  ;
4630                  ;
4640                  ;
4650                  ;           MODO PAGINA ZERO.Y
4660                  ;
4670                  ;
4680                  ;
4690   3AFE  20EB3A   ZERO.Y JSR ZEROPG
4700   3B01  20F639          JSR YINDEX
4710   3B04  A201            LDX #1
4720   3B06  A904            LDA #4
4730   3B08  60              RTS
4740                  ;
4750                  ;
4760                  ;
4770                  ;
4780                  ;
4790                  ;
4800                  ;
4810                  ;
4820                  ;
4830                  ;
4840                  ;
```

```
4850                    ;
4860                    ; ******************************************
4870                    ;
4880                    ;      UNO PSEUDO-MODO PER INSERIRE I TESTI
4890                    ;                    TXMODE
4900                    ;
4910                    ; ******************************************
4920                    ;
4930                    ;
4940                    ;
4950                    ;
4960                    ;
4970                    ;
4980                    ; LO PSEUDO-OPCODE TEX ($7F) E' L'INDICATORE
4990                    ; DI INIZIO TESTO, MENTRE LO PSEUDO-OPCODE
5000                    ; ETX ($FF) E' QUELLO DI FINE TESTO. TEX HA
5010                    ; UN MODO DI PSEUDO-INDIRIZZAMENTO: TXMODE.
5020                    ; IN TXMODE STAMPA LA STRINGA DI CARATTERI E
5030                    ; RITORNA SENZA OCCUPARSI DELLA LINEA IN ESA-
5040                    ; DECIMALE. LA STRINGA PUO' ESSERE DI QUALSIA-
5050                    ; SI LUNGHEZZA.
5060                    ;
5070                    ;
5080                    ;
5090                    ;
5100                    ;
5110                    ;
5120                    ;
5130                    ;
5140                    ;
5150                    ;
5160   3B09   68        TXMODE PLA              TOGLIE L'INDIRIZZO DI
5170   3B0A   68               PLA              RITORNO A OPERND.
5180                    ;
5190   3B0B   68               PLA              TOGLIE L'INDIRIZZO DI
5200   3B0C   68               PLA              RITORNO A DSLINE.
5210                    ;
5220                    ;
5230                    ;
5240                    ;
5250                    ;
5260   3B0D   207D37           JSR NEXTSL       PROSEGUE DOPO LO PSEUDO-
5270   3B10   300D             BMI TXFINE       OPCODE TEX.
5280   3B12   209532           JSR LEGBYT       LEGGE IL CARATTERE.
5290   3B15   C9FF             CMP #ETX         E' IL CARATTERE DI FINE?
5300   3B17   F006             BEQ TXFINE       SE SI' ESCE DAL LOOP,
5310   3B19   204034           JSR PR.CAR       SE NO, LO STAMPA.
5320   3B1C   18               CLC              LEGGE IL CARATTERE SUC-
5330   3B1D   90EE             BCC TXMODE+4     CESSIVO.
5340                    ;
5350                    ;
5360   3B1F   207234    TXFINE JSR CR.LF        LASCIA UNA RIGA.
5370   3B22   207D37           JSR NEXTSL       PASSA ALL'OPCODE SUCCESSIVO
5380   3B25   60               RTS              RITORNA ALLA ROUTINE CHE
5390                    ;                       HA CHIAMATO DSLINE.
5400                    ;
5410                    ;
5420                    ;
5430                    ;
5440                    ;
5450                    ;
5460                    ;
5470                    ;
5480                    ;
5490                    ; ******************************************
5500                    ;
```

```
5510                          ;      TABELLA DEI CODICI DI INDIRIZZAMENTO
5520                          ;
5530                          ; ******************************************
5540                          ;
5550                          ;
5560                          ;
5570.                         ;
5580                          ;
5590   3B26   783A          SUBS    .WORD IMPLID   IL MODO DI INDIRIZZAMENTO
5600                          ;                    O NON E' VALIDO, QUINDI
5610   3B28   6E3A                  .WORD ACC      IMPLICITO.
5620   3B2A   7D3A                  .WORD IMMEDT
5630   3B2C   EB3A                  .WORD ZEROPG
5640   3B2E   F33A                  .WORD ZERO.X
5650   3B30   FE3A                  .WORD ZERO.Y
5660   3B32   503A                  .WORD ABSLUT
5670   3B34   583A                  .WORD ABS.X
5680   3B36   633A                  .WORD ABS.Y
5690   3B38   783A                  .WORD IMPLID
5700   3B3A   BC3A                  .WORD RELATV
5710   3B3C   9D3A                  .WORD IND.X
5720   3B3E   AB3A                  .WORD IND.Y
5730   3B40   8F3A                  .WORD INDRCT
5740   3B42   093B                  .WORD TXMODE
```

CROSS REFERENCE

```
ABS.X   3A58     ABS.Y   3A63     ABSLUT  3A50     ACC     3A6E
CR      000D     CR.LF   3472     DEC.SL  331A     DSPAGE  3900
ETX     00FF     AVANTI  3AD1     LEGBYT  3295     HEX.PG  3500
IMMEDT  3A7D     IMPLID  3A78     INC.SL  330D     IND.X   3A9D
IND.Y   3AAB     INDRCT  3A8F     LF      000A     LPAREN  39E1
NEXTSL  377D     ONEBYT  39C8     REC.SL  352B     PR.ADR  359B
PR.BYT  3483     PR.CAR  3440     PRPAGE  3400     SAL.SL  3512
RELATV  3ABC     RELEND  3ADC     RPAREN  39E5     SELEZ   3205
SPAZIO  347D     SPAZI   3496     SUBS    3B26     TEX     007F
TWOBYT  39CF     TXFINE  3B1F     TXMODE  3B09     VISMON  3207
VMPAGE  3200     XINDEX  39EB     YINDEX  39F6     ZERO.X  3AF3
ZERO.Y  3AFE     ZEROPG  3AEB
```

C.8 Un disassemblatore (tabelle)

```
1000                    ;      APPENDICE C8: LISTATO ASSEMBLER DI
1010                    ;                    UN DISASSEMBLATORE
1020                    ;
1030                    ;                    TABELLE
1040                    ;
1050                    ;
1060                    ;
1070                    ;
1080                    ;               VEDI CAPITOLO 9
1090                    ;
1100                    ;
1110                    ;
1120                    ;
1130                    ;
1140                    ;
1150                    ;        COPYRIGHT (C) 1983 DI K. SKIER
1160                    ;           LEXINGTON, MASSACHUSETTS
1170                    ;
1180                    ;
1190                    ;
1200                    ;
1210                    ;
1220                    ;
1230                    ;
1240                    ;
1250                    ;
1260                    ;
1270                    ;
1280                    ;
1290                    ; ******************************************
1300                    ;
1310                    ;                   COSTANTI
1320                    ;
1330                    ; ******************************************
1340                    ;
1350                    ;
1360                    ;
1370                    ;
1380                    ;
1390                    ;
1400                    ;
1410                           TEX     = $7F
1420                    ;                INDICATORE DI INIZIO
1430                    ;                TESTO
1440                    ;
1450                    ;
1460                           ETX     = $FF
1470                    ;                INDICATORE DI FINE
1480                    ;                TESTO
1490                    ;
1500                    ;
1510                    ;
1520                    ;
1530                    ;
1540                    ;
1550                    ;
1560                    ;
1570                    ;
1580                    ;
1590                    ;
```

```
1600                        ;
1610                        ;
1620                        ;
1630                        ;
1640                        ;
1650                        ;
1660                        ;
1670                             DSPAGE = $3900
1680                        ;                          PAGINA DI PARTENZA DEL
1690                        ;                          DISASSEMBLATORE
1700                        ;
1710                        ;
1720                        ;
1730                        ;
1740                        ; ******************************************
1750                        ;
1760                        ;            LISTA DEGLI MNEMONICI
1770                        ;
1780                        ; ******************************************
1790                        ;
1800                        ;
1810                        ;
1820  0000   = 3B50              * =  DSPAGE+$250
1830                        ;
1840                        ;
1850                        ;
1860                        ;
1870                        ;
1880  3B50   7F             MNOMI  .BYTE TEX      SICCOME LA TABELLA E'
1890                        ;                     COMPOSTA DA STRINGHE DI
1900                        ;                     CARATTERI, PARTE CON LO
1910                        ;                     PSEUDO OPCODE 'TEX'
1920  3B51   455252              .BYTE 'ERR'
1930  3B54   414443              .BYTE 'ADC'
1940  3B57   414E44              .BYTE 'AND'
1950  3B5A   41534C              .BYTE 'ASL'
1960  3B5D   424343              .BYTE 'BCC'
1970  3B60   424353              .BYTE 'BCS'
1980  3B63   424551              .BYTE 'BEQ'
1990  3B66   424954              .BYTE 'BIT'
2000  3B69   424D49              .BYTE 'BMI'
2010  3B6C   424E45              .BYTE 'BNE'
2020  3B6F   42504C              .BYTE 'BPL'
2030  3B72   42524B              .BYTE 'BRK'
2040  3B75   425643              .BYTE 'BVC'
2050  3B78   425653              .BYTE 'BVS'
2060  3B7B   434C43              .BYTE 'CLC'
2070  3B7E   434C44              .BYTE 'CLD'
2080  3B81   434C49              .BYTE 'CLI'
2090  3B84   434C56              .BYTE 'CLV'
2100  3B87   434D50              .BYTE 'CMP'
2110  3B8A   435058              .BYTE 'CPX'
2120  3B8D   435059              .BYTE 'CPY'
2130  3B90   444543              .BYTE 'DEC'
2140  3B93   444558              .BYTE 'DEX'
2150  3B96   444559              .BYTE 'DEY'
2160  3B99   454F52              .BYTE 'EOR'
2170  3B9C   494E43              .BYTE 'INC'
2180  3B9F   494E58              .BYTE 'INX'
2190  3BA2   494E59              .BYTE 'INY'
2200  3BA5   4A4D50              .BYTE 'JMP'
2210  3BA8   4A5352              .BYTE 'JSR'
2220  3BAB   4C4441              .BYTE 'LDA'
2230  3BAE   4C4458              .BYTE 'LDX'
2240  3BB1   4C4459              .BYTE 'LDY'
2250  3BB4   4C5352              .BYTE 'LSR'
```

```
2260   3BB7  4E4F50              .BYTE 'NOP'
2270   3BBA  4F5241              .BYTE 'ORA'
2280   3BBD  504841              .BYTE 'PHA'
2290   3BC0  504850              .BYTE 'PHP'
2300   3BC3  504C41              .BYTE 'PLA'
2310   3BC6  504C50              .BYTE 'PLP'
2320   3BC9  524F4C              .BYTE 'ROL'
2330   3BCC  524F52              .BYTE 'ROR'
2340   3BCF  525449              .BYTE 'RTI'
2350   3BD2  525453              .BYTE 'RTS'
2360   3BD5  534243              .BYTE 'SBC'
2370   3BD8  534543              .BYTE 'SEC'
2380   3BDB  534544              .BYTE 'SED'
2390   3BDE  534549              .BYTE 'SEI'
2400   3BE1  535441              .BYTE 'STA'
2410   3BE4  535458              .BYTE 'STX'
2420   3BE7  535459              .BYTE 'STY'
2430   3BEA  544158              .BYTE 'TAX'
2440   3BED  544159              .BYTE 'TAY'
2450   3BF0  545358              .BYTE 'TSX'
2460   3BF3  545841              .BYTE 'TXA'
2470   3BF6  545853              .BYTE 'TXS'
2480   3BF9  545941              .BYTE 'TYA'
2490   3BFC  544558              .BYTE 'TEX'
2500                       ;
2510   3BFF  FF                  .BYTE ETX       FINE DELLA TABELLA.
2520                       ;
2530                       ;
2540                       ;
2550                       ;
2560                       ;
2570                       ;
2580                       ;
2590                       ;
2600                       ;
2610                       ;
2620                       ;
2630                       ;
2640                       ;
2650                       ;  ******************************************
2660                       ;
2670                       ;          TABELLA DEI CODICI MNEMONICI
2680                       ;
2690                       ;  ******************************************
2700                       ;
2710                       ;
2720                       ;
2730                       ;
2740                       ;
2750                       ;   IL CODICE DELLO MNEMONICO E'RAPPRESENTATO
2760                       ;     DALLA POSIZIONE NELLA TABELLA DEI NOMI
2770                       ;
2780                       ;
2790                       ;
2800   3C00  226A010101   MCODES .BYTE $22,$6A,1,1,1,$6A,$0A,1,$70
2810         6A0A0170
2820   3C09  6A0A01016A          .BYTE $6A,$0A,1,1,$6A,$0A,1
2830         0A01
2840   3C10  1F6A010101          .BYTE $1F,$6A,1,1,1,$6A,$0A,1
2850         6A0A01
2860   3C18  2B6A010101          .BYTE $2B,$6A,1,1,1,$6A,$0A,1
2870         6A0A01
2880   3C20  5807010116          .BYTE $58,7,1,1,$16,7,$79,1
2890         077901
2900   3C28  7607790116          .BYTE $76,7,$79,1,$16,7,$79,1
2910         077901
```

```
2920   3C30  1907010101         .BYTE $19,7,1,1,1,7,$79,1
2930         077901
2940   3C38  8807010101         .BYTE $88,7,1,1,1,7,$79,1
2950         077901
2960   3C40  7F49010101         .BYTE $7F,$49,1,1,1,$49,$64,1
2970         496401
2980   3C48  6D49640155         .BYTE $6D,$49,$64,1,$55,$49,$64,1
2990         496401
3000   3C50  2549010101         .BYTE $25,$49,1,1,1,$49,$64,1
3010         496401
3020   3C58  3149010101         .BYTE $31,$49,1,1,1,$49,$64,1
3030         496401
3040   3C60  8204010101         .BYTE $82,4,1,1,1,4,$7C,1
3050         047C01
3060   3C68  73047C0155         .BYTE $73,4,$7C,1,$55,4,$7C,1
3070         047C01
3080   3C70  2804010101         .BYTE $28,4,1,1,1,4,$7C,1
3090         047C01
3100   3C78  8E04010101         .BYTE $8E,4,1,1,1,4,$7C,$AC
3110         047CAC
3120   3C80  0191010197         .BYTE 1,$91,1,1,$97,$91,$94,1
3130         919401
3140   3C88  4601A30197         .BYTE $46,1,$A3,1,$97,$91,$94,1
3150         919401
3160   3C90  0D91010197         .BYTE $0D,$91,1,1,$97,$91,$94,1
3170         919401
3180   3C98  A991A30101         .BYTE $A9,$91,$A3,1,1,$91,1,1
3190         910101
3200   3CA0  615B5E0161         .BYTE $61,$5B,$5E,1,$61,$5B,$5E,1
3210         5B5E01
3220   3CA8  9D5B9A0161         .BYTE $9D,$5B,$9A,1,$61,$5B,$5E,1
3230         5B5E01
3240   3CB0  105B010161         .BYTE $10,$5B,1,1,$61,$5B,$5E,1
3250         5B5E01
3260   3CB8  345B9E0161         .BYTE $34,$5B,$9E,1,$61,$5B,$5E,1
3270         5B5E01
3280   3CC0  3D3701013D         .BYTE $3D,$37,1,1,$3D,$37,$40,1
3290         374001
3300   3CC8  523743013D         .BYTE $52,$37,$43,1,$3D,$37,$40,1
3310         374001
3320   3CD0  1C37010101         .BYTE $1C,$37,1,1,1,$37,$40,1
3330         374001
3340   3CD8  2E37010101         .BYTE $2E,$37,1,1,1,$37,$40,1
3350         374001
3360   3CE0  3A8501013A         .BYTE $3A,$85,1,1,$3A,$85,$4C,1
3370         854C01
3380   3CE8  4F8567013A         .BYTE $4F,$85,$67,1,$3A,$85,$4C,1
3390         854C01
3400   3CF0  1385010101         .BYTE $13,$85,1,1,1,$85,$4C,1
3410         854C01
3420   3CF8  8B85010101         .BYTE $8B,$85,1,1,1,$85,$4C,1
3430         854C01
3440                       ;
3450                       ;
3460                       ;
3470                       ;
3480                       ;
3490                       ;
3500                       ;
3510                       ;
3520                       ;
3530                       ;
3540                       ;
3550                       ; ********************************************
3560                       ;
3570                       ;     TABELLA DEI CODICI DI INDIRIZZAMENTO
```

```
3580                          ;
3590                          ; ****************************************
3600                          ;
3610                          ;
3620                          ;
3630                          ;
3640                          ; IL CODICE DI INDIRIZZAMENTO E'RAPPRESENTATO
3650                          ;     DALLA POSIZIONE NELLA TABELLA DELLE
3660                          ;        SUBROUTINE DI INDIRIZZAMENTO
3670                          ;
3680                          ;
3690                          ;
3700                          ;
3710                          ;
3720                          ;
3730  3D00  1216000000  MODI    .BYTE 18,22,0,0,0,6,6,0
3740        060600
3750  3D08  1204020000          .BYTE 18,4,2,0,0,12,12,0
3760        0C0C00
3770  3D10  1418000000          .BYTE 20,24,0,0,0,14,14,0
3780        0E0E00
3790  3D18  1210000000          .BYTE 18,16,0,0,0,22,22,0
3800        161600
3810  3D20  0C16000006          .BYTE 12,22,0,0,6,6,6,0
3820        060600
3830  3D28  120402000C          .BYTE 18,4,2,0,12,12,12,0
3840        0C0C00
3850  3D30  1418000000          .BYTE 20,24,0,0,0,8,8,0
3860        080800
3870  3D38  1210000000          .BYTE 18,16,0,0,0,14.14,0
3880        0E0E00
3890  3D40  1216000000          .BYTE 18,22,0,0,0,6,6,0
3900        060600
3910  3D48  120C02000C          .BYTE 18,12,2,0,12,12,12,0
3920        0C0C00
3930  3D50  1418000000          .BYTE 20,24,0,0,0,8,8,0
3940        080800
3950  3D58  1210000000          .BYTE 18,16,0,0,0,14,14,0
3960        0E0E00
3970  3D60  1216000000          .BYTE 18,22,0,0,0,6,6,0
3980        060600
3990  3D68  120402001A          .BYTE 18,4,2,0,26,12,12,0
4000        0C0C00
4010  3D70  1418000000          .BYTE 20,24,0,0,0,8,8,0
4020        080800
4030  3D78  1210000000          .BYTE 18,16,0,0,0,14,14,28
4040        0E0E1C
4050                          ;
4060  3D80  0016000006          .BYTE 0,22,0,0,6,6,6,0
4070        060600
4080  3D88  120012000C          .BYTE 18,0,18,0,12,12,12,0
4090        0C0C00
4100  3D90  1418000008          .BYTE 20,24,0,0,8,8,10,0
4110        080A00
4120  3D98  1210120000          .BYTE 18,16,18,0,0,14,0,0
4130        0E0000
4140  3DA0  0416040006          .BYTE 4,22,4,0,6,6,6,0
4150        060600
4160  3DA8  120412000C          .BYTE 18,4,18,0,12,12,12,0
4170        0C0C00
4180  3DB0  1418000008          .BYTE 20,24,0,0,8,8,10,0
4190        080A00
4200  3DB8  141012000E          .BYTE 20,16,18,0,14,14,16,0
4210        0E1000
4220  3DC0  0416000006          .BYTE 4,22,0,0,6,6,6,0
4230        060600
```

```
4240   3DC8  120412000C          .BYTE 18,4,18,0,12,12,12,0
4250         0C0C00
4260   3DD0  1418000000          .BYTE 20,24,0,0,0,8,8,0
4270         080800
4280   3DD8  1210000000          .BYTE 18,16,0,0,0,14,14,0
4290         0E0E00
4300   3DE0  0416000006          .BYTE 4,22,0,0,6,6,6,0
4310         060600
4320   3DE8  120412000C          .BYTE 18,4,18,0,12,12,12,0
4330         0C0C00
4340   3DF0  1418000000          .BYTE 20,24,0,0,0,8,8,0
4350         080800
4360   3DF8  1210000000          .BYTE 18,16,0,0,0,14,14,0
4370         0E0E00
```

```
                    CROSS REFERENCE


DSPAGE 3900      ETX    00FF      MCODES 3C00      MNOMI   3B50
MODI   3D00      TEX    007F
```

C.9 Utility di trasferimento

```
1000   ;        APPENDICE C9: LISTATO ASSEMBLER DELLE
1010   ;                      UTILITY DI TRASFERIMENTO
1020   ;
1030   ;
1040   ;
1050   ;              VEDI CAPITOLO 10
1060   ;
1070   ;
1080   ;
1090   ;
1100   ;
1110   ;           COPYRIGHT (C) 1983 DI K. SKIER
1120   ;              LEXINGTON, MASSACHUSETTS
1130   ;
1140   ;
1150   ;
1160   ;
1170   ;
1180   ;
1190   ;
1200   ; ******************************************
1210   ;
1220   ;                  COSTANTI
1230   ;
1240   ; ******************************************
1250   ;
1260   ;
1270   ;
1280   ;
1290   ;
1300           CR      = $0D   RITORNO CARRELLO
1310           LF      = $0A   'A CAPO'
1320           TEX     = $7F   INDICATORE DI INIZIO TESTO
1330           ETX     = $FF   INDICATORE DI FINE TESTO
1340   ;
1350   ;
1360   ;
1370   ;
1380   ;
1390   ;
1400   ; ******************************************
1410   ;
1420   ;             INDIRIZZI ESTERNI
1430   ;
1440   ; ******************************************
1450   ;
1460   ;
1470   ;
1480   ;
1490   ;
1500   ;
1510   ;
1520   ;
1530           VMPAGE  = $3200
1540                           PAGINA DI PARTENZA DEL
1550   ;                       MONITOR VISIBILE.
1560   ;
1570           SELEZ   = VMPAGE+5
1580           VISMON  = VMPAGE+7
1590   ;
1600   ;
1610   ;
```

```
1620                             PRPAGE   = $3400
1630                                              PAGINA DI PARTENZA DELLE
1640                        ;                     UTILITY DI STAMPA.
1650                             TVT.ON   = PRPAGE+8
1660                             INVIA:   = PRPAGE+$E4
1670                             SAL.SL   = PRPAGE+$112
1680                             REC.SL   = PRPAGE+$12B
1690                        ;
1700                        ;
1710                             HEX.PG   = $3500
1720                                              INDIRIZZO DI PARTENZA DEL
1730                        ;                     DUMP ESADECIMALE. L'INDI-
1740                        ;                     RIZZO EFFETTIVO E'$3550
1750                        ;                     MA E' PIU' FACILE CONTARE
1760                        ;                     DA $3500.
1770                        ;
1780                             SETADS   = HEX.PG+$E3
1790                        ;
1800                        ;
1810                        ;
1820                        ;
1830                        ;
1840                        ;
1850                        ;
1860                        ;
1870                        ; ******************************************
1880                        ;
1890                        ;                 VARIABILI
1900                        ;
1910                        ; ******************************************
1920                        ;
1930                        ;
1940                        ;
1950                        ;
1960                        ;
1970   0000   = 37B0             * = $37B0
1980                        ;
1990                        ;
2000                             SA       = HEX.PG+$52
2010                                              PUNTATORE DELL'INDIRIZZO
2020                        ;                     DI PARTENZA DEL BLOCCO DA
2030                        ;                     MUOVERE.
2040                             EA       = SA+2
2050                                              PUNTATORE DELL'INDIRIZZO
2060                        ;                     DI FINE BLOCCO DA MUOVERE
2070                        ;
2080   37B0   0000          NUM  .WORD 0          NUMERO DI BYTE DEL BLOCCO
2090                        ;                     DA MUOVERE. ZERO SIGNIFI-
2100                        ;                     CA CHE IL BLOCCO CONTIENE
2110                        ;                     UN BYTE.
2120                        ;
2130   37B2   0000          DEST .WORD 0          PUNTATORE DELL'INDIRIZZO
2140                        ;                     DI DESTINAZIONE DEL BLOC-
2150                        ;                     CO.
2160                        ;
2170                        ;
2180                        ;
2190                        ;
2200                        ;
2210                        ;
2220                                          PUNTATORI DI PAGINA: LEGGO-
2230                                          NO E SCRIVONO I BYTE:
2240                               LEGPTR   = $FB
2250                               SCRPTR   = LEGPTR+2
2260                        ;
2270                        ;
```

```
2280                            ;
2290                            ;
2300                            ;
2310                            ;
2320                            ;
2330                            ;
2340                            ;  ***************************************
2350                            ;
2360                            ;            UTILITY DI TRASFERIMENTO
2370                            ;
2380                            ;  ***************************************
2390                            ;
2400                            ;
2410                            ;
2420                            ;
2430                            ;
2440                            ;
2450                            ;
2460   37B4   200834            MOVER  JSR TVT.ON       ATTIVA LO SCHERMO.
2470   37B7   20E434                   JSR INVIA:       VISUALIZZA IL TITOLO.
2480   37BA   7F0D0A                   .BYTE TEX,CR,LF
2490   37BD   2020202020               .BYTE '      TRANSFER.'     ;
2500          5452414E53
2510          4645522E
2520   37CC   0D0A0AFF                 .BYTE CR,LF,LF,ETX
2530                            ;
2540   37D0   20E335                   JSR SETADS       LEGGE L'INDIRIZZO DI PAR-
2550                            ;                       TENZA E QUELLO DI FINE.
2560                            ;
2570   37D3   20B938                   JSR SET.DA       INDIRIZZO DI DESTINAZIONE
2580                            ;                       AVENDO SETTATO QUESTI
2590                            ;                       PUNTATORI POSSIAMO ESE-
2600                            ;                       GUIRE MOV.EA:
2610                            ;
2620                            ;
2630                            ;
2640                            ;
2650                            ;
2660                            ;
2670                            ;
2680                            ;  ***************************************
2690                            ;
2700                            ;       MOV.EA: TRASFERISCE IL BLOCCO
2710                            ;               SPECIFICATO DA SA, EA,DEST
2720                            ;  ***************************************
2730                            ;
2740                            ;
2750                            ;
2760                            ;
2770                            ;
2780                            ;  CODICI DI RITORNO:
2790                            ;
2800                            ;
2810                                   ERROR   = 0      CODICE DI RITORNO:NON SI
2820                            ;                       PUO' MUOVERE PERCHE'SA>EA
2830                                   OKAY    = $FF    CODICE DI RITORNO PER
2840                            ;                       TRASFERIMENTO ESEGUITO
2850                            ;
2860                            ;
2870   37D6   AE5535            MOV.EA LDX EA+1         SETTA NUM = EA - SA:
2880   37D9   38                       SEC
2890   37DA   AD5435                   LDA EA
2900   37DD   ED5235                   SBC SA
2910   37E0   8DB037                   STA NUM
2920   37E3   B002                     BCS MOVE.1
2930   37E5   CA                       DEX
```

```
2940   37E6  38                SEC
2950   37E7  8A         MOVE.1 TXA
2960   37E8  ED5335            SBC SA+1
2970   37EB  8DB137            STA NUM+1
2980   37EE  B003              BCS MOVNUM      NUM = EA - SA
2990                    ;
3000   37F0  A900       ER.RTN LDA #ERROR      SE EA < SA,
3010   37F2  60                RTS             TORNA CON CODICE ERRORE.
3020                    ;
3030                    ;
3040                    ;
3050                    ;
3060                    ; ********************************************
3070                    ;
3080                    ;    MOVNUM: TRASFERISCE IL BLOCCO
3090                    ;            SPECIFICATO DA SA, NUM, DEST
3100                    ; ********************************************
3110                    ;
3120                    ;
3130                    ;
3140   37F3  A003       MOVNUM LDY #3          SALVA I 4 BYTE DI PAGINA
3150   37F5  B9FB00     SAVE   LDA LEGPTR,Y    ZERO CHE USEREMO.
3160   37F8  48                PHA
3170   37F9  88                DEY
3180   37FA  10F9              BPL SAVE
3190                    ;
3200                    ;
3210   37FC  38                SEC             DESTINAZIONE < ORIGINE?
3220   37FD  AD5335            LDA SA+1
3230   3800  CDB337            CMP DEST+1
3240   3803  9040              BCC MOVSU
3250   3805  D018              BNE MOVGIU
3260                    ;                      SE SI' MUOVE IN GIU'
3270                    ;
3280   3807  AD5235            LDA SA
3290   380A  CDB237            CMP DEST
3300   380D  9036              BCC MOVSU
3310   380F  D00E              BNE MOVGIU
3320   3811  A000       OK.RTN LDY #0          RESETTA I BYTE USATI IN
3330                    ;                      PAGINA ZERO.
3340   3813  68         RESET  PLA
3350   3814  99FB00            STA LEGPTR,Y
3360   3817  C8                INY
3370   3818  C004              CPY #4
3380   381A  D0F7              BNE RESET
3390   381C  A9FF              LDA #OKAY       TORNA COL CODICE 'OKAY'.
3400   381E  60                RTS
3410                    ;
3420                    ;
3430                    ;
3440   381F  20A438     MOVGIU JSR LOPAG       SETTA I PUNTATORI ALLA
3450                    ;                      PAGINA PIU' BASSA DI
3460                    ;                      ORIGINE E DI ARRIVO.
3470                    ;
3480                    ;
3490   3822  A000              LDY #0          Y DEVE ESSERE UGUALE A 0
3500                    ;                      IN OGNI CASO.
3510                    ;
3520   3824  AEB137            LDX NUM+1       C'E' PIU' DI UNA PAGINA
3530                    ;                      DA MUOVERE?
3540                    ;
3550   3827  F00E              BEQ MENGIU      SE NO, MUOVE MENO DI UNA
3560                    ;                      PAGINA.
3570                    ;
3580                    ;
3590   3829  B1FB       PAGGIU LDA (LEGPTR),Y LEGGE UN BYTE DAL BLOCCO
```

```
3600   382B   91FD              STA (SCRPTR),Y  DI ORIGINE E LO SCRIVE
3605                       ;                    NEL BLOCCO DI DESTINAZIONE
3610   382D   C8                INY             E' L'ULTIMO BYTE DELLA
3620   382E   D0F9              BNE PAGGIU      PAGINA?
3630                       ;
3640                       ;
3650   3830   E6FC              INC LEGPTR+1    INCREMENTA I PUNTATORI.
3660   3832   E6FE              INC SCRPTR+1
3670   3834   CA                DEX             ANCORA PIU' DI UNA PAGINA
3675                       ;                    DA MUOVERE?
3680   3835   D0F2              BNE PAGGIU      SE SI', MUOVE UN'ALTRA
3690                       ;                    PAGINA.
3700                       ;
3710   3837   88           MENGIU DEY
3720   3838   C8                INY             MUOVE MENO DI UNA PAGINA
3730   3839   B1FB              LDA (LEGPTR),Y  PARTENDO DAL BASSO.
3740   383B   91FD              STA (SCRPTR),Y
3750   383D   CCB037            CPY NUM         E' L'ULTIMO BYTE?
3760   3840   D0F6              BNE MENGIU+1    SE NO, MUOVE IL PROSSIMO.
3770   3842   4C1138            JMP OK.RTN      SE SI' TORNA CON CODICE
3780                       ;                    'OKAY'.
3790                       ;
3800                       ;
3810                       ;
3820   3845   ADB137       MOVSU  LDA NUM+1     C'E' PIU' DI UNA PAGINA
3825                       ;                    DA MUOVERE?
3830   3848   F048              BEQ MENOSU      SE NO, MUOVE MENO DI UNA
3840                       ;                    PAGINA.
3850                       ;                    PER MUOVERE PIU' DI UNA
3860                       ;                    PAGINA SETTA I PUNTATO-
3870                       ;                    RI DI PAGINA LEGPTR E
3880                       ;                    SCRPTR ALLA PAGINA PIU'
3890                       ;                    ALTA DEL BLOCCO DI ORIGI-
3900                       ;                    NE E DESTINAZIONE. PERCIO'
3910                       ;                    BISOGNA USARE X COME BYTE
3920                       ;                    ALTO E Y COME BYTE BASSO
3930                       ;                    DI UN PUNTATORE CHE CHIA-
3940                       ;                    MEREMO (X,Y). PRIMA DI
3950                       ;                    TUTTO SETTIAMO (X,Y).
3960                       ;                    (X,Y) = NUM - $FF
3970                       ;                    CHE E' L'INDIRIZZO RELA-
3980                       ;                    TIVO DELLA PAGINA PIU'
3990                       ;                    ALTA DEL BLOCCO.
4000   384A   ACB137            LDY NUM+1       Y CONTIENE IL BYTE ALTO
4010   384D   ADB037            LDA NUM         DELLA DIMENSIONE DEL BLOC-
4015                       ;                    CO, A QUELLO BASSO.
4020   3850   38                SEC             PREPARA LA SOTTRAZIONE
4030   3851   E9FF              SBC #$FF
4040   3853   B001              BCS NEXT.1
4050   3855   88                DEY
4060
4070   3856   AA           NEXT.1 TAX           ORA (X,Y) = NUM - $FF
4080                       ;                    X E' IL BYTE BASSO E Y
4090                       ;                    IL BYTE ALTO DI NUM-$FF.
4100                       ;
4110                       ;
4120                       ;
4130   3857   84FE              STY SCRPTR+1
4140   3859   8A                TXA
4150   385A   18                CLC             PREPARA PER L'ADDIZIONE
4160   385B   6D5235            ADC SA
4170   385E   85FB              STA LEGPTR
4180   3860   9001              BCC NEXT.2
4190   3862   C8                INY
4200                       ;
4210                       ;
```

```
4220   3863  98          NEXT.2 TYA
4230   3864  6D5335             ADC SA+1
4240   3867  85FC               STA LEGPTR+1
4250                     ;
4260                     ;                         ORA LEGPTR = SA+NUM-$FF.
4270                     ;                         (L'ULTIMA PAGINA DEL
4280                     ;                         BLOCCO DI ORIGINE).
4290                     ;
4300   3869  8A                 TXA
4310   386A  18                 CLC                PREPARA PER L'ADDIZIONE
4320   386B  6DB237             ADC DEST
4330   386E  85FD               STA SCRPTR
4340   3870  9002               BCC NEXT.3
4350   3872  E6FE               INC SCRPTR+1
4360                     ;
4370                     ;
4380   3874  A5FE        NEXT.3 LDA SCRPTR+1
4390   3876  6DB337             ADC DEST+1
4400   3879  85FE               STA SCRPTR+1       SCRPTR = DEST+NUM-$FF
4410                     ;                         QUINDI LEGPTR E SCRPTR
4420                     ;                         PUNTANO RISPETTIVAMENTE
4430                     ;                         ALL'ULTIMA PAGINA DEL
4440                     ;                         BLOCCO DI ORIGINE E DI
4450                     ;                         DESTINAZIONE.
4460   387B  AEB137             LDX NUM+1          X = NUMERO DI PAGINE DA
4470                     ;                         MUOVERE.
4480                     ;
4490   387E  A0FF        PAGSU  LDY #$FF           MUOVE UNA PAGINA.
4500                     ;
4510
4520   3880  B1FB        SULOOP LDA (LEGPTR),Y     LEGGE UN BYTE.
4530   3882  91FD               STA (SCRPTR),Y     SCRIVE UN BYTE.
4540   3884  88                 DEY                SETTA L'INDICE PER IL
4550                     ;                         PROSSIMO BYTE.
4560                     ;
4570   3885  D0F9               BNE SULOOP         ESEGUE DI NUOVO SE NON
4580                     ;                         ERA L'ULTIMO BYTE
4590                     ;
4600                     ;
4610                     ;
4620   3887  B1FB               LDA (LEGPTR),Y     MUOVE L'ULTIMO BYTE.
4630   3889  91FD               STA (SCRPTR),Y
4640   388B  C6FC               DEC LEGPTR+1       DECREMENTA IL PUNTATORE
4650   388D  C6FE               DEC SCRPTR+1       DI PAGINA.
4660   388F  CA                 DEX                ANCORA PIU' DI UNA PAGI-
4665                     ;                         NA DA MUOVERE?
4670   3890  D0EC               BNE PAGSU          SE SI' MUOVE UN'ALTRA
4680                     ;                         PAGINA.
4690                     ;
4700                     ;
4710   3892  20A438      MENOSU JSR LOPAG          SETTA LEGPTR E SCRPTR.
4720   3895  ACB037             LDY NUM            SETTA L'INDICE AL NUMERO
4730                     ;                         DI BYTE DA MUOVERE.
4740   3898  B1FB        POCSU  LDA (LEGPTR),Y     MUOVE UN BYTE
4750   389A  91FD               STA (SCRPTR),Y
4760   389C  88                 DEY                DECREMENTA L'INDICE
4770   389D  C0FF               CPY #$FF           COPIATO L'ULTIMO BYTE?
4780   389F  D0F7               BNE POCSU          SE NO, NE MUOVE UN ALTRO.
4790   38A1  4C1138             JMP OK.RTN         SE SI' RITORNA CON IL CO-
4800                     ;                         DICE 'OKAY'.
4810                     ;
4820                     ;
4830                     ;
4840                     ;
4850                     ;
4860                     ;
```

```
4870                          ;
4880                          ;
4890                          ;
4900                          ;
4910                          ;
4920                          ; *****************************************
4930                          ;
4940                          ;       SETTA I PUNTATORI DI PAGINA AL BASSO
4950                          ;       DEI BLOCCHI DI ORIGINE E DESTINAZIONE
4960                          ;
4970                          ; *****************************************
4980                          ;
4990                          ;
5000                          ;
5010                          ;
5020                          ;
5030   38A4  AD5235           LOPAG  LDA SA
5040   38A7  85FB                    STA LEGPTR
5050   38A9  AD5335                  LDA SA+1
5060   38AC  85FC                    STA LEGPTR+1
5070                          ;
5080                          ;
5090   38AE  ADB237                  LDA DEST
5100   38B1  85FD                    STA SCRPTR
5110   38B3  ADB337                  LDA DEST+1
5120   38B6  85FE                    STA SCRPTR+1
5130                          ;
5140                          ;
5150   38B8  60                      RTS
5160                          ;
5170                          ;
5180                          ;
5190                          ;
5200                          ;
5210                          ;
5220                          ;
5230                          ; *****************************************
5240                          ;
5250                          ;         INDIRIZZO DI DESTINAZIONE
5260                          ;
5270                          ; *****************************************
5280                          ;
5290                          ;
5300                          ;
5310                          ;
5320                          ;
5330                          ;
5340                          ;
5350                          ;
5360                          ;
5370                          ;
5380                          ;
5390                          ;
5400                          ;
5410                          ;
5420   38B9  200834           SET.DA JSR TVT.ON      ATTIVA IL VIDEO.
5430   38BC  20E434                  JSR INVIA:
5440   38BF  7F0D0A                  .BYTE TEX,CR,LF                    ;
5450   38C2  494E444952              .BYTE 'INDIRIZZO DI DESTINAZIONE'  ;
5460         495A5A4F20              .BYTE '(PREMI Q)'
5470         4449204445
5480         5354494E41
5490         5A494F4E45
5500         2028505245
5505         4D49205129
5510   38DE  FF                      .BYTE ETX
```

```
5520   38DF   200732           JSR VISMON     RICHIAMA IL M.V. PERCHE'
5530                   ;                      L'UTENTE POSSA SPECIFI-
5540   38E2   AD0532   DESQUI  LDA SELEZ      CARE L'INDIRIZZO DI PAR-
5550   38E5   8DB237           STA DEST       TENZA.
5560   38E8   AD0632           LDA SELEZ +1   SETTA L'INDIRIZZO DI DE-
5570   38EB   8DB337           STA DEST+1     STINAZIONE UGUALE A QUEL-
5580                   ;                      LO SPECIFICATO DALL'UTEN-
5590   38EE   60               RTS            TE E RITORNA.
```

CROSS REFERENCE

```
CR      000D    DESQUI  38E2    DEST    37B2    EA      3554
ER.RTN  37F0    ERROR   0000    ETX     00FF    LEGPTR  00FB
HEX.PG  3500    MENGIU  3837    MENOSU  3892    LF      000A
SAVE    37F5    RESET   3813    SULOOP  3880    LOPAG   38A4
MOV.EA  37D6    MOVE.1  37E7    POCSU   3898    MOVGIU  381F
MOVER   37B4    MOVSU   3845    MOVNUM  37F3    NEXT.1  3856
NEXT.2  3863    NEXT.3  3874    NUM     37B0    OK.RTN  3811
OKAY    00FF    PAGGIU  3829    PAGSU   387E    REC.SL  352B
INVIA   34E4    PRPAGE  3400    SAL.SL  3512    SCRPTR  00FD
SA      3552    SELEZ   3205    SET.DA  38B9    SETADS  35E3
TEX     007F    TVT.ON  3408    VISMON  3207    VMPAGE  3200
```

C.10 Un semplice text editor (primo livello e subroutine di visualizzazione)

```
1000          ;    APPENDICE C10: LISTATO ASSEMBLER DI
1010          ;         UN SEMPLICE TEXT EDITOR
1020          ;     (PRIMO LIVELLO E SUBROUTINE DI
1030          ;           VISUALIZZAZIONE)
1040          ;
1050          ;
1060          ;
1070          ;           VEDI CAPITOLO 11
1080          ;
1090          ;
1100          ;
1110          ;
1120          ;
1130          ;
1140          ;        COPYRIGHT (C) 1983 DI K. SKIER
1150          ;          LEXINGTON, MASSACHUSETTS
1160          ;
1170          ;
1180          ;
1190          ;
1200          ;
1210          ;
1220          ;
1230          ;
1240          ;
1250          ;
1260          ;
1270          ;
1280          ;
1290          ; ********************************************
1300          ;
1310          ;               COSTANTI
1320          ;
1330          ; ********************************************
1340          ;
1350          ;
1360          ;
1370          ;
1380          ;
1390               CR     = $0D   RITORNO CARRELLO
1400          ;
1410               LF     = $0A   'A CAPO'
1420          ;
1430               TEX    = $7F   INDICATORE DI INIZIO
1440          ;                   TESTO.
1450          ;
1460               ETX    = $FF   INDICATORE DI FINE
1470          ;                   TESTO.
1480          ;
1490               INSCAR = 'I'   CARATTERE DI INSERIMENTO
1500               SOSCAR = 'O'   CARATTERE DI SOSTITUZIO-
1510          ;                   NE.
1520          ;
1530          ;
1540          ;
1550          ;
1560          ;
1570          ;
1580          ;
1590          ; ********************************************
```

```
1600                    ;           INDIRIZZI ESTERNI
1610                    ;
1620                    ;
1630                    ; *******************************************
1640                    ;
1650                    ;
1660                    ;
1670                         TV.PTR = $FB     PUNTATORE DELLA POSIZIO-
1680                         PARAMS = $3000   NE SULLO SCHERMO DEI DA-
                                              TI DI SISTEMA.
1690                    ;
1700                    ;
1710                         TVCOL  = PARAMS+3
1720                         TVRIG  = PARAMS+4
1730                         FRECC  = PARAMS+7
1740                    ;
1750                    ;
1760                    ;
1770                         TVSUBS = $3100
1780                         PULXY  = TVSUBS+$13
1790                         TVHOME = TVSUBS+$2B
1800                         TVTOXY = TVSUBS+$3C
1810                         TVGIU  = TVSUBS+$76
1820                         TVVAI  = TVSUBS+$7F
1830                         TVPIU  = TVSUBS+$81
1840                         TV.SCR = TVSUBS+$9B
1850                         VEDBYT = TVSUBS+$A3
1860                         TVSALV = TVSUBS+$C4
1870                         TV.POP = TVSUBS+$D3
1880                    ;
1890                    ;
1900                         VMPAGE = $3200
                                             PAGINA DI PARTENZA
1910                    ;                    DEL MONITOR VISIBILE.
1920                    ;
1930                         SELEZ  = VMPAGE+5
1940                         LEGBYT = VMPAGE+$95
1950                         INC.SL = VMPAGE+$10D
1960                         DEC.SL = VMPAGE+$11A
1970
1980                    ;
1990                    ;
2000                         PRPAGE = $3400
                                             PAGINA DI PARTENZA DELLE
2010                                         UTILITY DI STAMPA.
2020                    ;
2030
2040                         TVT.ON = PRPAGE+8
2050                         TVTOFF = PRPAGE+$0E
2060                         PR.ON  = PRPAGE+$14
2070                         PR.OFF = PRPAGE+$1A
2080                         PR.CAR = PRPAGE+$40
2090                         INVIA: = PRPAGE+$E4
2100                         SAL.SL = PRPAGE+$112
2110                         POP.SL = PRPAGE+$12B
2120                    ;
2130                    ;
2140                         HEX.PG = $3500
                                             INDIRIZZO DI PARTENZA DEL
2150                                         DUMP ESADECIMALE.
2160                    ;
2170                    ;
2180                         SA     = HEX.PG+$52
2190                         EA     = SA+2
2200                         SETADS = HEX.PG+$E3
2210                         NEXTSL = HEX.PG+$27D
2220                         GOTOSA = HEX.PG+$29A
2230                    ;
2240                    ;
2250                         EDPAGE = $3E00
```

```
2260                                          PAGINA DI PARTENZA
2270                        ;                 DELL'EDITOR.
2280                             EDITA  = EDPAGE+$C8
2290                        ;
2300                        ;
2310                        ;
2320                        ;
2330                        ;
2340                        ;
2350                        ; ****************************************
2360                        ;
2370                        ;               VARIABILI
2380                        ;
2390                        ; ****************************************
2400                        ;
2410                        ;
2420                        ;
2430  0000   = 3E00                *  =  EDPAGE
2440                        ;
2450                        ;
2460                        ;
2470  3E00   00             CONTAT .BYTE 0      CONTATORE USATO DA LINE.2
2480  3E01   00             EDMODO .BYTE 0      FLAG: 0 PER SOSTITUZIONE
2490                        ;                         1 PER INSERIMENTO.
2500                        ;
2510                        ;
2520                        ;
2530                        ; ****************************************
2540                        ;
2550                        ;      TEXT EDITOR: PRIMO LIVELLO
2560                        ;
2570                        ; ****************************************
2580                        ;
2590                        ;
2600                        ;
2610                        ;
2620                        ;
2630                        ;
2640  3E02   200F3E         EDITOR JSR SETBUF     INIZIALIZZA LE VARIABILI
2645                        ;                     E I PUNTATORI.
2650  3E05   20373E         EDLOOP JSR MOSTRA     VISUALIZZA UNA PORZIONE
2660                        ;                     DEL BUFFER.
2670  3E08   20C83E                JSR EDITA      PERMETTE LA CORREZIONE O
2680                        ;                     IL MOVIMENTO NEL BUFFER.
2690  3E0B   18                    CLC
2700  3E0C   18                    CLC
2710  3E0D   90F6                  BCC EDLOOP
2720                        ;
2730                        ;
2740                        ;
2750                        ;
2760                        ;
2770                        ;
2780                        ;
2790                        ;
2800                        ;
2810                        ;
2820                        ; ****************************************
2830                        ;
2840                        ;    INIZIALIZZA I PUNTATORI DEL BUFFER
2850                        ;
2860                        ; ****************************************
2870                        ;
2880                        ;
2890                        ;
2900                        ;
```

```
2910  3E0F  200834      SETBUF JSR TVT.ON       ATTIVA IL VIDEO
2920  3E12  20E434             JSR INVIA:      VISUALIZZA IL TITOLO
2930  3E15  7F0D0A0A           .BYTE TEX,CR,LF,LF
2940  3E19  494E495A49         .BYTE 'INIZIALIZZAZIONE BUFFER.' ;
2950        414C495A5A
2960        415A494F4E
2970        4520425546
2975        4645522E
2980  3E2C  0D0A0AFF           .BYTE CR,LF,LF,ETX
2990  3E30  20E335             JSR SETADS      LEGGE GLI INDIRIZZI SPE-
3000                    ;                      CIFICATI DALL'UTENTE.
3010  3E33  209A37             JSR GOTOSA      SELEZ PUNTA ALL'INIZIO
3020                    ;                      DEL BUFFER
3030  3E36  60                 RTS             RITORNA.
3040                    ;
3050                    ;
3060                    ;
3070                    ;
3080                    ;
3090                    ;
3100                    ;
3110                    ; ******************************************
3120                    ;
3130                    ;                      MOSTRA
3140                    ;
3150                    ; ******************************************
3160                    ;
3170                    ;
3180                    ;
3190                    ;
3200                    ;
3210  3E37  20C431      MOSTRA JSR TVSALV      SALVA I BYTE DI PAG ZERO
3220                    ;
3230  3E3A  202B31             JSR TVHOME      SETTA LA POSIZIONE HOME
3240                    ;
3250                    ;
3260                    ;
3270  3E3D  AE0330             LDX TVCOL       PULISCE 3 RIGHE DEL VIDEO
3280  3E40  A003               LDY #3
3290  3E42  201331             JSR PULXY
3300                    ;
3310                    ;
3320  3E45  202B31             JSR TVHOME
3330                    ;
3340  3E48  207631             JSR TVGIU       PASSA ALLA RIGA 2
3350  3E4B  20C431             JSR TVSALV
3360  3E4E  205E37             JSR LINE.2      VISUALIZZA IL TESTO DELLA
3370                    ;                      RIGA DUE.
3380                    ;
3390  3E51  20D331             JSR TVREC
3400  3E54  207631             JSR TVGIU       PASSA ALLA RIGA 3
3410                    ;
3420  3E57  [illegible]        JSR LINE.3      VISUALIZZA IL TESTO DELLA
3430                    ;                      RIGA TRE
3440                    ;
3450  3E5A  20D331             JSR TVREC       RESETTA I BYTE DI PAGINA
3460  3E5D  60                 RTS             E RITORNA
3470                    ;
3480                    ;
3490                    ;
3500                    ;
3510                    ;
3520                    ;
3530                    ;
3540                    ;
3550                    ;
```

```
3560                     ; ****************************************
3570                     ;
3580                     ;       VISUALIZZA LA LINEA DI TESTO
3590                     ;
3600                     ; ****************************************
3610                     ;
3620                     ;
3630                     ;
3640                     ;
3650                     ;
3660  3E5E  201235       LINE.2 JSR SAL.SL     SALVA IL PUNTATORE SELEZ
3670  3E61  AD0330              LDA TVCOL
3680  3E54  4A                  LSR A
3690  3E65  AA                  TAX           X = TVCOL/2
3700  3E66  CA                  DEX
3710                     ;
3720  3E67  201A33       LOOP.1 JSR DEC.SL     DECREMENTA SELEZ X VOLTE
3730  3E6A  CA                  DEX
3740  3E6B  10FA                BPL LOOP.1
3750                     ;
3760  3E6B  AD0330              LDA TVCOL     INIZIALIZZA IL CONTATORE
3770  3E70  8D003E              STA CONTAT    PER VISUALIZZARE TVCOL
3780                     ;                    CARATTERI
3790  3E73  209532       LOOP.2 JSR LEGBYT     LEGGE UN CARATTERE DAL
3795                     ;                    BUFFER
3800  3E76  209B31              JSR TV.SCR    LO VISUALIZZA SUL VIDEO
3810  3E79  207F31              JSR TVVAI     .SELEZIONA LA LOCAZIONE
3815                     ;                    SUCCESSIVA SUL VIDEO
3820  3E7C  200D33              JSR INC.SL    SELEZIONA IL BYTE SUC-
3830                     ;                    CESSIVO DEL BUFFER
3840  3E7F  CE003E              DEC CONTAT    E' L'ULTIMO CARATTERE ?
3850  3E82  10EF                BPL LOOP.2    SE NO PASSA AL SUCCESSIVO
3860                     ;
3870                     ;
3880  3E84  202B35              JSR REC.SL    RESETTA SELEZ DALLO STACK
3890  3E87  60                  RTS           RITORNA
3900                     ;
3910                     ;
3920                     ;
3930                     ;
3940                     ;
3950                     ; ****************************************
3960                     ;
3970                     ;       VISUALIZZA LA LINEA DI STATO
3980                     ;
3990                     ; ****************************************
4000                     ;
4010                     ;
4020                     ;
4030                     ;
4040                     ;
4050  3E88  AD0330       LINE.3 LDA TVCOL
4060  3E8B  4A                  LSR A         A = TVCOL/2
4070  3E8C  E902                SBC #2        A = (TVCOL/2)-2
4080  3E8E  208131              JSR TVPIU     ORA TV.PTR PUNTA DUE CA-
4090                     ;                    RATTERI PIU' A SINISTRA
4100                     ;                    DEL CENTRO DELLA LINEA 3
4110                     ;
4120  3E91  AD013E              LDA EDMODO    QUALE MODO E' SELEZIONATO?
4130  3E94  C901                CMP #1        MODO INSERIMENTO?
4140  3E96  D005                BNE SOMODO    SE NO IL MODO SOSTITUZIONE
4150                     ;
4160  3E98  A949                LDA #INSCAR   SE SI' METTE IN A IL CA-
4170  3E9A  18                  CLC           RATTERE DI INSERIMENTO
4180  3E9B  9002                BCC TVMODO
4190  3E9D  A94F         SOMODO LDA #SOSCAR   METTE IN A IL CARATTERE
```

```
                                         DI SOSTITUZIONE
4195                         ;
4200   3E9F   209B31         TVMODO JSR TV.SCR   VISUALIZZA IL CARATTERE
4210   3EA2   A902                  LDA #2       DI MODO
4220   3EA4   208131                JSR TVPIU    ORA TV.PTR PUNTA AL CENTRO
4230                         ;                   DELLA LINEA 3
4240                         ;
4250   3EA7   AD0730                LDA FRECC    VISUALIZZA LA FRECCIA
4260   3EAA   209B31                JSR TV.SCR   INDICE
4270                         ;
4280   3EAD   A902                  LDA #2       ORA TV.PTR PUNTA ALLA PO-
4290   3EAF   208131                JSR TVPIU    SIZIONE RISERVATA ALL'IN-
4300                         ;                   DIRIZZO DEL CARATTERE
4310                         ;                   CORRENTE
4320   3EB2   AD0632                LDA SELEZ +1 VISUALIZZA L'INDIRIZZO
4330   3EB5   20A331                JSR VEDBYT
4340   3EB8   AD0532                LDA SELEZ
4350   3EBB   20A331                JSR VEDBYT
4360                         ;
4370   3EBE   60                    RTS          RITORNA
```

CROSS REFERENCE

```
FRECC   3007    PULXY   3113    CONTAT  3E00    CR      000D
DEC.SL  331A    EA      3554    EDITA   3EC8    EDITOR  3E02
EDLOOP  3E05    EDMODO  3E01    EDPAGE  3E00    ETX     00FF
LEGBYT  3295    GOTOSA  379A    HEX.PG  3500    INC.SL  330D
INSCAR  0049    LF      000A    LINE.2  3E5E    LINE.3  3E88
LOOP.1  3E67    LOOP.2  3E73    NEXTSL  377D    SOMODO  3E9D
SOSCAR  004F    PARAMS  3000    REC.SL  352B    PR.CAR  3440
PR.OFF  341A    PR.ON   3414    INVIA:  34E4    PRPAGE  3400
SAL.SL  3512    SA      3552    SELEZ   3205    SETADS  35E3
SETBUF  3E0F    MOSTRA  3E37    TEX     007F    TVREC   31D3
TV.PTR  00FB    TV.SCR  319B    TVCOL   3003    TVGIU   3176
TVHOME  312B    TVMODO  3E9F    TVPIU   3181    TVSALV  31C4
TVRIG   3004    TVVAI   317F    TVSUBS  3100    TVT.ON  3408
TVTOFF  340E    TVTOXY  313C    VMPAGE  3200    VEDBYT  31A3
```

C.11 Un semplice text editor (subroutine EDITA)

```
1000                    ;     APPENDICE C11: LISTATO ASSEMBLER DI
1010                    ;           UN SEMPLICE TEXT EDITOR
1020                    ;              SUBROUTINE EDITA
1030                    ;
1040                    ;
1050                    ;
1060                    ;
1070                    ;               VEDI CAPITOLO 11
1080                    ;
1090                    ;
1100                    ;
1110                    ;
1120                    ;
1130                    ;
1140                    ;          COPYRIGHT (C) 1983 DI K. SKIER
1150                    ;             LEXINGTON, MASSACHUSETTS
1160                    ;
1170                    ;
1180                    ;
1190                    ;
1200                    ;
1210                    ;
1220                    ;
1230                    ;
1240                    ;
1250                    ;
1260                    ;
1270                    ;
1280                    ;
1290                    ; ****************************************
1300                    ;
1310                    ;                 COSTANTI
1320                    ;
1330                    ; ****************************************
1340                    ;
1350                    ;
1360                    ;
1370                    ;
1380                    ;
1390                          CR      = $0D    RITORNO CARRELLO
1400                    ;
1410                          LF      = $0A    'A CAPO'
1420                    ;
1440                          TEX     = $7F    INDICATORE DI INIZIO
1450                    ;                      TESTO.
1460                    ;
1470                          ETX     = $FF    INDICATORE DI FINE
1480                    ;                      TESTO.
1490                    ;
1500                    ;
1510                    ;
1520                    ;
1530                    ;
1540                    ;
1550                    ;
1560                    ;
1570                    ;
1580                    ; ****************************************
1590                    ;
1600                    ;            INDIRIZZI ESTERNI
```

```
1610                      ;
1620                      ;  ******************************************
1630                      ;
1640                      ;
1650                      ;
1660                      ;
1670                      ;
1680                            VMPAGE = $3200
                                              PAGINA DI PARTENZA
1690                      ;                   DEL MONITOR VISIBILE
1700                      ;
1710                            SELEZ  = VMPAGE+5
1720                            VISMON = VMPAGE+7
1730                            LEGBYT = VMPAGE+$95
1740                            LEGTAS = VMPAGE+$E0
1750                            INC.SL = VMPAGE+$10D
1760                            DEC.SL = VMPAGE+$11A
1770                            MEM.SL = VMPAGE+$12D
1780                      ;
1790                      ;
1800                            PRPAGE = $3400
                                              PAGINA DI PARTENZA DELLE
1810                      ;                   UTILITY DI STAMPA
1820                      ;
1830                      ;
1840                            PR.ON  = PRPAGE+$14
1850                            PR.OFF = PRPAGE+$1A
1860                            INVCAR = PRPAGE+$40
1870                            SAL.SL = PRPAGE+$112
1880                            REC.SL = PRPAGE+$12B
1890                      ;
1900                      ;
1910                            HEX.PG = $3500
                                              INDIRIZZO DI PARTENZA DEL
1920                      ;                   DUMP ESADECIMALE
1930                      ;
1940                      ;
1950                            SA     = HEX.PG+$52
1960                            EA     = SA+2
1970                            SAQUI  = HEX.PG+$161
1980                            NEXTSL = HEX.PG+$27D
1990                            GOTOSA = HEX.PG+$29A
2000                      ;
2010                      ;
2020                            MOVERS = $37B0
                                              INDIRIZZO DELL'UTILITY
2030                      ;                   DI TRASFERIMENTO
2040                      ;
2050                            DEST   = MOVERS+2
2060                            MOV.EA = MOVERS+$26
2070                            DESQUI = MOVERS+$132
2080                      ;
2090                            EDPAGE = $3E00
                                              PAGINA DI PARTENZA
2100                      ;                   DELL'EDITOR
2110                      ;
2120                            EDCOM  = EDPAGE+$C0
2130                      ;
2140                      ;
2150                      ;
2160                      ;
2170                      ;
2180                      ;
2190                      ;  ******************************************
2200                      ;
2210                      ;               VARIABILI
2220                      ;
2230                      ;  ******************************************
2240                      ;
2250                      ;
2260                      ;
```

```
2270                          EDMODO = EDPAGE+1  0 PER SOSTITUZIONE
2280                    ;                        1 PER INSERIMENTO
2290                    ;
2300   0000  = 3EC0             * = EDCOM
2310                    ;
2320                    ;         TASTI DI FUNZIONE
2330                    ;
2340                    ;                       L'EDITOR RICONOSCE QUEI
2350                    ;                       TASTI COME TASTI DI FUN-
2360                    ;                       ZIONE.ASSEGNA A UN TASTO
2370                    ;                       UNA FUNZIONE MEMORIZZAN-
2380                    ;                       DO IL SUO CODICE IN UNO
2390                    ;                       DEI SEGUENTI DATA-BYTE:
2400                    ;
2410                    ;
2420                    ;
2430   3EC0  93         TELIM  .BYTE $93        QUESTO TASTO PULISCE IL
2440                    ;                       BUFFER. $93 E' IL TASTO
2450                    ;                       "CLR".
2460                    ;
2470                    ;
2480                    ;
2490   3EC1  94         TMODO  .BYTE $94        QUESTO TASTO CAMBIA IL
2500                    ;                       MODO DELL'EDITOR DA IN-
2510                    ;                       SERIMENTO A SOSTITUZIO-
2520                    ;                       NE E VICEVERSA.$94 E'IL
2530                    ;                       TASTO "INS".
2540                    ;
2550   3EC2  1D         TSUCC  .BYTE $1D        QUESTO TASTO SELEZIONA
2560                    ;                       IL PROSSIMO CARATTERE
2570                    ;                       NEL BUFFER. $1D E' IL
2580                    ;                       TASTO "->" CHE SERVE
2590                    ;                       PER SPOSTARSI VERSO DE-
2600                    ;                     . STRA.
2610                    ;
2620   3EC3  9D         TPREC  .BYTE $9D        QUESTO TASTO SELEZIONA
2630                    ;                       IL PRECEDENTE CARATTERE
2640                    ;                       NEL BUFFER. $9D E' IL
2650                    ;                       TASTO "<-" CHE SERVE
2660                    ;                       PER SPOSTARSI VERSO SI-
2670                    ;                       NISTRA.
2680                    ;
2690   3EC4  10         TPRT   .BYTE $10        STAMPA IL BUFFER. $10
2700                    ;                       E' IL CODICE CORRISPON-
2710                    ;                       DENTE A CONTROL - P.
2720                    ;
2730                    ;
2740   3EC5  14         TCANC  .BYTE $14        CODICE DEL TASTO "CAN-
2750                    ;                       CELLAZIONE".CANCELLA
2760                    ;                       IL CARATTERE SELEZIONATO
2770                    ;
2780                    ;
2790   3EC6  51         TFINE  .BYTE 'Q'        PREMENDO DUE VOLTE IL
2800                    ;                       TASTO "FINE",L'EDITOR
2810                    ;                       RITORNA ALLA SUBROUTINE
2820                    ;                       CHE LO HA RICHIAMATO.
2830                    ;
2840                    ;
2850                    ;
2860                    ;
2870                    ;                       ALTRE VARIABILI:
2880                    ;
2890   3EC7  00         TEMPCA .BYTE 0          BYTE USATI DA EDITA.
2900                    ;
2910                    ;
2920                    ;
```

```
2930                             ;
2940                             ;
2950                             ;
2960                             ;
2970                             ;
2980                             ;
2990                             ;
3000                             ; *******************************************
3010                             ;
3020                             ;   TEXT EDITOR: SUBROUTINE DI AGGIORNAMENTO
3030                             ;
3040                             ; *******************************************
3050                             ;
3060                             ;
3070                             ;
3080                             ;
3090                             ;
3100                             ;
3110                             ;
3120  3EC8  20E032     EDITA   JSR LEGTAS    LEGGE IL VALORE DEL TASTO
                                             PREMUTO.
3130                             ;
3140  3ECB  CDC63E             CMP TFINE     E' IL TASTO "FINE"?
3150  3EC3  D017               BNE TFAI      SE NO,ESEGUE LA FUNZIONE
                                             RICHIESTA.
3160                             ;
3170                             ;
3180  3ED0  48                 PHA           SALVA IL VALORE DEL TASTO
3190  3ED1  20E032             JSR LEGTAS    NELLO STACK.SE L'UTENTE
                                             HA DIGITATO DUE VOLTE IL
3200                             ;
3210  3ED4  CDC63E             CMP TFINE     TASTO FINE DEVE USCIRE
                                             DALL'EDITOR.
3215                             ;
3220  3ED7  D004               BNE NOFINE    SE NO,DEVE COMPIERE LE
                                             FUNZIONI RICHIESTE RI-
3230                             ;           SPETTANDO L'ORDINE.
3240                             ;
3250                             ;
3260  3ED9  68         FINEED  PLA           USCITA DALL'EDITOR:LEVA
                                             DALLO STACK IL VALORE
3270                             ;
3280  3EDA  68                 PLA           DEL TASTO "FINE".LEGGE
3290  3ED8  68                 PLA           L'INDIRIZZO DI RITORNO
3300  3EDC  60                 RTS           AL PRIMO LIVELLO E RI-
                                             TORNA.
3310                             ;
3320  3EDD  8DC73E     NOFINE  STA TEMPCA    SALVA IL TASTO CHE HA
                                             SEGUITO QUELLO DI "FINE".
3330                             ;
3340  3EE0  68                 PLA           LEGGE DALLO STACK IL PRI-
                                             MO TASTO "FINE".
3345                             ;
3350  3EE1  20E73E             JSR TFAI      ESEGUE LA FUNZIONE RI-
                                             CHIESTA.
3355                             ;
3360  3EE4  ADC73E             LDA TEMPCA    RIPORTA NELL'ACCUMULATORE
                                             IL SECONDO TASTO "FINE".
3370                             ;
3380                             ;
3390                             ;
3400                             ;
3410                             ;
3420  3EE7  CDC13E     TFAI    CMP TMODO     E' IL TASTO DI CAMBIO MO-
3430  3EEA  D00B               BNE IFSUCC    DO? SE NO ESEGUE IL TEST
                                             SUCCESSIVO.
3435                             ;
3440  3EEC  CE013E             DEC EDMODO    SE SI' CAMBIA MODO
3450  3EEF  1005               BPL FINEFA
3460  3EF1  A901               LDA #1
3470  3EF3  8D013E             STA EDMODO
3480  3EF6  60         FINEFA  RTS           RITORNA.
3490                             ;
3500                             ;
3510  3EF7  CDC23E     IFSUCC  CMP TSUCC     E' IL TASTO "SUCCESSIVO"?
3520  3EFA  D004               BNE IFPREC    SE NO ESEGUE IL TEST
                                             SUCCESSIVO.
3530                             ;
3540  3EFC  20793F             JSR NEXTCA    SE SI AVANZA IL CURSORE
```

```
3550                      ;                        DI UNA POSIZIONE
3560   3EFF  60                  RTS               E RITORNA.
3570                      ;
3580                      ;
3590   3F00  CDC33E       IFPREC CMP TPREC         E' IL TASTO "PRECEDENTE"?
3600   3F03  D004                BNE IFCANC        SE NO,ESEGUE IL TEST
3605                      ;                        SUCCESSIVO.
3610   3F05  20873F              JSR PRECCA        SE SI,RETROCEDE CON IL
3620   3F08  60                  RTS               CURSORE DI UNA POSIZIONE
3630                      ;                        E RITORNA.
3640                      ;
3650   3F09  CDC53E       IFCANC CMP TCANC         TASTO "CANCELLAZIONE"?
3660   3F0C  D004                BNE IFPRT         (CARATTERE)
3670   3F0E  20DD3F              JSR CANCEL        SE SI',CANCELLA IL CARAT-
3680   3F11  60                  RTS               TERE ATTUALE E RITORNA.
3690                      ;
3700                      ;
3710   3F12  CDC43E       IFPRT  CMP TPRT          E' IL TASTO "STAMPA"?
3720   3F15  D004                BNE IFELIM
3730   3F17  20C53F              JSR PRTBUF        SE SI',STAMPA IL CONTENUTO
3740   3F1A  60                  RTS               DEL BUFFER E RITORNA.
3750                      ;
3760                      ;
3770                      ;
3780   3F1B  CDC03E       IFELIM CMP TELIM         E' IL TASTO "ELIMINA"?
3790   3F1E  D004                BNE TCAR
3800                      ;
3810   3F20  20B43F              JSR ELIM          SE SI' ELIMINA IL TESTO
3820   3F23  60                  RTS               CONTENUTO NEL BUFFER E
3830                      ;                        RITORNA.
3840                      ;
3850                      ;
3860                      ;
3870                      ; OK.NON E' UNA FUNZIONE DELL'EDITOR, BENSI'
3880                      ; UN CARATTERE. A SECONDA DEL MODO IN CUI SIA-
3890                      ; MO, POSSIAMO INSERIRE O SOSTITUIRE UN CARAT-
3900                      ; TERE.
3910                      ;
3920   3F24  AE013E       TCAR   LDX EDMODO        E' IN MODO SOSTITUZIONE?
3930   3F27  F004                BEQ RIBAT         SE SI', SOSTITUISCE IL
3940                      ;                        CARATTERE.
3950   3F29  20343F              JSR INSER         SE NO, LO INSERISCE.
3960   3F2C  60                  RTS               RITORNA.
3970                      ;
3980   3F2D  202D33       RIBAT  JSR MEM.SL        METTE IL CARATTERE NELLA
3990                      ;                        CORRENTE LOCAZIONE.
4000   3F30  207D37              JSR NEXTSL        SELEZIONA IL PROSSIMO
4010   3F33  60                  RTS               CARATTERE E RITORNA.
4020                      ;
4030                      ;
4040                      ;
4050                      ;
4060                      ;
4070   3F34  48           INSER  PHA               SALVA IL CARATTERE DA
4080                      ;                        INSERIRE
4090                      ;
4100   3F35  201235              JSR SAL.SL        SALVA L'INDIRIZZO DEL CA-
4105                      ;                        RATTERE SELEZIONATO.
4110   3F38  AD5335              LDA SA+1          SALVA LA LOCAZIONE DI
4120   3F3D  48                  PHA               PARTENZA DEL BUFFER.
4130   3F3C  AD5235              LDA SA
4140   3F3F  48                  PHA
4150                      ;
4160                      ;
4170   3F40  AD5535              LDA EA+1          SALVA LA LOCAZIONE DI
4180   3F43  48                  PHA               FINE DEL BUFFER.
```

```
4190   3F44  AD5435          LDA EA
4200   3F47  48              PHA
4210                   ;
4220                   ;
4230   3F48  206136          JSR SAQUI      SA = SELEZ
4240                   ;
4250                   ;
4260                   ;
4270                   ;
4280                   ;
4290   3F4B  207D37          JSR NEXTSL     AVANZA ALLA SUCCESSIVA
                                            POSIZIONE DEL BUFFER.
4300                   ;
4310   3F4E  3011            BMI FINEIN     SE FINE BUFFER SOSTITUI-
                                            SCE.
4320                   ;
4330                   ;
4340                   ;
4350                   ;
4360   3F50  20E238          JSR DESQUI     DEST = SELEZ
4370                   ;
4380                   ;
4390                   ;
4400                   ;
4410                   ;
4420   3F53  AD5435          LDA EA         DECREMENTA L'INDIRIZZO DI
4430   3F56  D003            BNE DEC.EA     FINE PER MUOVERE IL TESTO
4440   3F58  CE5535          DEC EA+1       OLTRE LA FINE DEL BUFFER.
4450   3F5B            . DEC.EA
4460   3F5B  CE5435          DEC EA
4470                   ;
4480                   ;
4490                   ;
4500   3F5E  20D637    UNOSU JSR MOV.EA     CREA SPAZIO PER UN BYTE
                                            ALLA CORRENTE LOCAZIONE
4510                   ;                    MUOVENDO A DEST IL BLOCCO
4520                   ;                    SPECIFICATO DA EA E SA.
4530                   ;
4540                   ;
4550                   ;
4560   3F61  68        FINEIN PLA           RESETTA EA
4570   3F62  8D5435          STA EA
4580   3F65  68              PLA
4590   3F66  8D5535          STA EA+1
4600                   ;
4610                   ;
4620   3F69  68              PLA            RESETTA SA
4630   3F6A  8D5235          STA SA
4640   3F6D  68              PLA
4650   3F6E  8D5335          STA SA+1
4660                   ;
4670                   ;
4680   3F71  202B35          JSR REC.SL     RESETTA SELEZ
4690                   ;
4700                   ;
4710                   ;
4720   3F74  68              PLA            RESETTA L'ACCUMULATORE CON
                                            IL CARATTERE DA INSERIRE.
4730                   ;
4740                   ;
4750   3F75  202D3F          JSR RIBAT
4760   3F78  60              RTS            RITORNA.
4770   3F79  209532    NEXTCA JSR LEGBYT    LEGGE IL CARATTERE SELE-
                                            ZIONATO.
4775                   ;
4780   3F7C  C9FF            CMP #ETX       E' ETX?
4790   3F7E  F004            BEQ AN.ETX     SE SI', RITORNA CON CODICE
                                            "MENO".
4800                   ;
4810                   ;
4820   3F80  207D37          JSR NEXTSL     SE NO SELEZIONA IL PROSSI-
                                            MO BYTE DEL BUFFER
4830                   ;
```

```
4840   3F83  60                  RTS           E RITORNA CON CODICE "PIU".
4850                    ;
4860                    ;
4870                    ;
4880   3F84  A9FF       AN.ETX LDA #$FF
4890   3F86  60                  RTS
4900                    ;
4910                    ;
4920                    ;
4930                    ;
4940                    ;
4950   3F87  38         PRECCA SEC             PREPARA LA COMPARAZIONE
4960   3F88  AD5335            LDA SA+1
4970   3F8B  CD0632            CMP SELEZ+1
4980   3F8E  900C              BCC SL.OK       SE SI' SELEZ PUO' ESSERE
4990   3F90  D010              BNE NO.OK       DECREMENTATO.
5000                    ;
5010                    ;
5020                    ;
5030   3F97  AD5235            LDA SA
5040   3F95  CD0532            CMP SELEZ
5050   3F98  F017              BEQ NO.DEC      SE SELEZ = SA NON DECRE-
5060                    ;                      MENTA.
5070   3F9A  B006              BCS NO.OK
5080                    ;
5090   3F9C  201A33     SL.OK  JSR DEC.SL      SE SELEZ E' OK DECREMENTA
5100                    ;
5110                    ;
5120   3F9F  A900              LDA #0          GENERA CODICE "PIU'"
5130   3FA1  60                RTS             E RITORNA.
5140                    ;
5150                    ;
5160   3FA2  AD5235     NO.OK  LDA SA          SE SELEZ<SA LO PORTA A
5165                    ;                      UN VALORE ACCETTABILE.
5170   3FA5  8D0532            STA SELEZ       SELEZ = SA
5180   3FA8  AD5335            LDA SA+1
5190   3FAB  8D0632            STA SELEZ+1  .
5200   3FAE  A900              LDA #0
5210   3FB0  60                RTS             RITORNA.
5220                    ;
5230                    ;
5240   3FB1  A9FF       NO.DEC LDA #$FF        SELEZ = SA QUINDI NON
5245                    ;                      DECREMENTA
5250   3FB3  60                RTS             E RITORNA CON CODICE
5260                    ;                      "MENO".
5270                    ;
5280                    ;
5290                    ;
5300   3FB4  209A37     ELIM   JSR GOTOSA      SELEZ = PRIMA POSIZIONE
5305                    ;                      DEL BUFFER.
5310   3FB7  A9FF       ELOOP  LDA #ETX        METTE ETX NELLA POSIZIONE
5320   3FB9  202D33            JSR MEM.SL      SELEZIONATA.
5330   3FBC  207D37            JSR NEXTSL      SELEZIONA LA POSIZIONE
5340                    ;                      SUCCESSIVA.
5350   3FBF  10F6              BPL ELOOP       SE NON E' L'ULTIMA POSI-
5360                    ;                      ZIONE, RIPETE.
5370                    ;
5380                    ;
5390   3FC1  209A37            JSR GOTOSA      SE HA FINITO RESETTA
5400                    ;                      SELEZ ALL'INIZIO.
5410                    ;
5420   3FC4  60                RTS             RITORNA.
5430   3FC5  209A37     PRTBUF JSR GOTOSA      SELEZ PUNTA ALL'INIZIO
5435                    ;                      DEL BUFFER.
5440   3FC8  201434            JSR PR.ON       SELEZIONA LA STAMPANTE.
5450   3FCB  209532     PRLOOP JSR LEGBYT      LEGGE IL CARATTERE SELE-
```

```
                                              ZIONATO.
5455                    ;                     E' ETX?
5460  3FCE  C9FF               CMP #ETX       SE SI' FINISCE DI STAMPARE
5470  3FD0  F008               BEQ FINEPR     SE NO INVIA IL CARATTERE
5480  3FD2  204034             JSR INVCAR     ALLE PERIFERICHE ATTIVATE
5485                    ;                     SELEZIONA IL CARATTERE
5490  3FD5  207D37             JSR NEXTSL     SUCCESSIVO.
5495                    ;                     SE NON E' FINE BUFFER
5500  3FD8  10F1               BPL PRLOOP     PROSEGUE.
5510                    ;
5520                    ;
5530  3FDA  4C1A34      FINEPR JMP PR.OFF
5540                    ;
5550                    ;
5560                    ;
5570                    ;
5580                    ;                     SALVA L'INDIRIZZO DEL CA-
5590  3FDD  201235      CANCEL JSR SAL.SL     RATTERE SELEZIONATO.
5595                    ;                     SALVA L'INDIRIZZO DI INI-
5600  3FE0  AD5335             LDA SA+1       ZIO BUFFER.
5610  3FE3  48                 PHA
5620  3FE4  AD5235             LDA SA
5630  3FE7  48                 PHA
5640                    ;                     SETTA DEST = SELEZ PERCHE'
5650  3FE8  20E238             JSR DESQUI     DEVE MUOVERE UN BLOCCO
5660                    ;                     DI TESTO.
5670                    ;
5680                    ;
5690                    ;                     AVANZA DI UN CARATTERE
5700  3FEB  207D37             JSR NEXTSL     NEL BUFFER SE POSSIBILE.
5710                    ;                     SETTA SA = SELEZ PERCHE'
5720  3FEE  206136             JSR SAQUI      IL BLOCCO DA MUOVERE PAR-
5730                    ;                     TE 1 BYTE SOPRA IL CARAT-
5740                    ;                     TERE SELEZIONATO.
5750                    ;
5760                    ;
5770                    ;                     MUOVE IL BLOCCO SPECIFI-
5780  3FF1  20D637             JSR MOV.EA     CATO DA SA, EA E DEST.
5790                    ;
5800                    ;
5810                    ;                     RESETTA SA.
5820  3FF4  68                 PLA
5830  3FF5  8D5235             STA SA
5840  3FF8  68                 PLA
5850  3FF9  8D5335             STA SA+1
5860  3FFC  202B35             JSR REC.SL     RESETTA SELEZ.
5870  3FFF  60                 RTS            RITORNA.
```

CROSS REFERENCE

AN.ETX	3F84	TCAR	3F24	CR	000D	DESQUI	38E8
DEC.EA	3F5B	DEC.SL	331A	CANCEL	3FDD	DEST	37B2
FINEFA	3EF6	TFAI	3EE7	EA	3554	EDITA	3EC8
EDCOM	3EC0	EDMODO	3E01	EDPAGE	3E00	FINEED	3ED9
FINEIN	3F61	FINEPR	3FDA	ETX	00FF	ELOOP	3FB7
TELIM	3EC0	ELIM	3FB4	LEGBYT	3295	LEGTAS	32E0
GOTOSA	379A	HEX.PG	3500	IFPRT	3F12	IFCANC	3F09
IFELIM	3F1B	IFSUCC	3EF7	IFPREC	3F00	INC.SL	330D
INSER	3F34	LF	000A	TMODO	3EC1	MOV.EA	37D6
MOVERS	37B0	NEXTCA	3F79	TSUCC	3EC2	NEXTSL	377D
NO.DEC	3FB1	NO.OK	3FA2	NOFINE	3EDD	UNOSU	3F5E
REC.SL	352B	INVCAR	3440	PR.OFF	341A	PR.ON	3414

```
TPREC  3EC3   PRECCA 3F87   PRLOOP 3FCB   PRPAGE 3400
PRTBUF 3FC5   TPRT   3EC4   SAL.SL 3512   MEM.SL 332D
TFINE  3EC6   TCANC  3EC5   SA     3552   SAQUI  3661
SELEZ  3205   SL.OK  3F9C   RIBAT  3F2D   TEMPCA 3EC7
TEX    007F   VISMON 3207   VMPAGE 3200
```

C.12 Il Monitor Visibile integrato

```
1000                    ;      APPENDICE C12: LISTATO ASSEMBLER DEL
1010                    ;          MONITOR VISIBILE INTEGRATO
1020                    ;
1030                    ;
1040                    ;
1050                    ;
1060                    ;              VEDI CAPITOLO 12
1070                    ;
1080                    ;
1090                    ;
1100                    ;
1110                    ;
1120                    ;
1130                    ;
1140                    ;         COPYRIGHT (C) 1983 DI K. SKIER
1150                    ;          LEXINGTON, MASSACHUSETTS
1160                    ;
1170                    ;
1180                    ;
1190                    ;
1200                    ;
1210                    ;
1220                    ;
1230                    ;
1240                    ;
1250                    ;
1260                    ;
1270                    ;
1280                    ;
1290                    ;
1300                    ;
1310                    ;
1320                    ;
1330                    ;
1340                    ;
1350                    ;
1360                    ;
1370                    ; ******************************************
1380                    ;
1390                    ;            INDIRIZZI ESTERNI
1400                    ;
1410                    ; ******************************************
1420                    ;
1430                    ;
1440                    ;
1450                    ;
1460                    ;
1470                    ;
1480       PRPAGE = $3400
1490                    ;          PAGINA DI PARTENZA DELLE
1500                    ;          UTILITY DI STAMPA.
1510                    ;
1520       PRINTR = PRPAGE
1530       USR.FN = PRPAGE+2
1540                    ;
1550                    ;
1560       HEX.PG = $3500
1570                    ;          INDIRIZZO DELLA PAGINA DI
1580                    ;          PARTENZA DEL DUMP ESADECI-
1590                    ;          MALE.
1600       TVDUMP = HEX.PG+$57
1610       PRDUMP = HEX.PG+$A8
```

```
1620                     ;
1630                     ;
1640                             DSPAGE = $3900
1650                     ;                        PAGINA DI PARTENZA DEL
1660                     ;                        DISASSEMBLER
1670                             TV.DIS = DSPAGE+9
1680                             PR.DIS = DSPAGE+$26
1690                     ;
1700                             MOVERS = $37B0
1710                     ;                        PAGINA DI PARTENZA DEL-
1720                     ;                        L'UTILITY DI TRASFERI-
1725                     ;                        MENTO.
1730                             MOVER  = MOVERS+4
1740                     ;
1750                     ;
1760                             EDPAGE = $3E00
1770                     ;                        PAGINA DI PARTENZA DEL
1780                     ;                        TEXT EDITOR.
1790
1800                             EDITOR = EDPAGE+2
1810                     ;
1820                     ;
1830                     ;
1840                     ;
1850                     ;
1860                     ;
1870                     ;
1880                     ;
1890  0000  = 30B0               * =  $30B0
1900                     ;
1910                     ;
1920                     ; ****************************************
1930                     ;
1940                     ;      AMPLIAMENTI DEL MONITOR VISIBILE
1950                     ;
1960                     ; ****************************************
1970                     ;
1980                     ;
1990                     ;
2000  30B0  C950         ESPAND CMP #'P'          E' IL TASTO 'P' ?
2010  30B2  D009                BNE IF.U          SE NO, ESEGUE IL TEST SUC-
2020  30B4  AD0034              LDA PRINTR        CESSIVO SE SI' ATTIVA LA
2030  30B7  49FF                EOR #$FF          STAMPANTE
2040  30B9  8D0034              STA PRINTR
2050  30BC  60                  RTS               E RITORNA.
2060                     ;
2070  30BD  C955         IF.U   CMP #'U'          E' IL TASTO 'U' ?
2080  30BF  D009                BNE IF.H          SE NO, ESEGUE IL TEST
2090  30C1  AD0234              LDA USR.FN        SUCCESSIVO,SE SI' ATTIVA
2100  30C4  49FF                EOR #$FF          LA ROUTINE UTENTE
2110  30C6  8D0234              STA USR.FN
2120  30C9  60                  RTS               E RITORNA.
2130                     ;
2140  30CA  C948         IF.H   CMP #'H'          E' IL TASTO 'H' ?
2150  30CC  D00D                BNE IF.M          SE NO, ESEGUE IL TEST
2160  30CE  AD0034              LDA PRINTR        E' ATTIVA LA STAMPANTE ?
2170  30D1  D004                BNE NEXT.1        SE SI' STAMPA IL DUMP
2180  30D3  205735              JSR TVDUMP        SE NO, LO VISUALIZZA
2190  30D6  60                  RTS               E RITORNA.
2200                     ;
2210  30D7  20A835       NEXT.1 JSR PRDUMP        STAMPA IL DUMP ...
2220  30DA  60                  RTS               ...E RITORNA.
2230                     ;
2240  30DB  C94D         IF.M   CMP #'M'          E' IL TASTO 'M' ?
2250  30DD  D004                BNE IF.DIS        SE NO ESEGUE IL TEST
2260  30DF  20B437              JSR MOVER         SUCCESSIVO SE SI' CHIEDE
```

```
2270   30E2  60                  RTS           GLI INDIRIZZI E MUOVE IL
                                               BLOCCO SPECIFICATO.
2280                     ;
2290   30E3  C93F        IF.DIS  CMP #'?'      E' IL TASTO '?' ?
2300   30E5  D00D                BNE IF.T      SE NO, ESEGUE IL TEST
2310   30E7  AD0034              LDA PRINTR    E' ATTIVA LA STAMPANTE ?
2320   30EA  D004                BNE NEXT.2    SE SI' STAMPA IL DISAS-
2330   30EC  200939              JSR TV.DIS    SEMBLATO SE NO LO VISUA-
2340   30EF  60                  RTS           LIZZA E RITORNA.
2350                     ;
2360   30F0  202639      NEXT.2  JSR PR.DIS    STAMPA IL DISASSEMBLATO
2370   30F3  60                  RTS           E RITORNA.
2380                     ;
2390   30F4  C954        IF.T    CMP #'T'      E' IL TASTO 'T' ?
2400   30F6  D004                BNE FINE      SE NO, RITORNA.
2410   30F8  20023E              JSR EDITOR    SE SI' RICHIAMA IL
2420   30FB  60                  RTS           TEXT EDITOR E RITORNA.
2430                     ;
2440   30FC  60          FINE    RTS           PER INTEGRARE MAGGIORMEN-
2450                     ;                     TE IL MONITOR VISIBILE
2460                     ;                     SOSTITUIRE QUESTO 'RTS'
2470                     ;                     CON UN 'JMP' AL PROGRAM-
2480                     ;                     MA CHE ESEGUE ULTERIORI
2490                     ;                     TEST E SALTI.

                     CROSS REFERENCE

DSPAGE 3900     EDITOR 3E02     EDPAGE 3E00     EXIT   30F
ESPAND 30B0     HEX.PG 3500     IF.DIS 30E3     IF.H   30C
IF.M   30DB     IF.T   30F4     IF.U   30BD     MOVER  37B
MOVERS 37B0     NEXT.1 30D7     NEXT.2 30F0     PR.DIS 392
PRDUMP 35A8     PRINTR 3400     PRPAGE 3400     TV.DIS 390
TVDUMP 3557     USR.FN 3402
```

C.13 Dati di sistema per il VIC-20

```
1000                ;       APPENDICE C13: LISTATO ASSEMBLER DEL
1010                ;             BLOCCO DEI DATI DI SISTEMA
1020                ;                   PER IL VIC-20
1030                ;
1040                ;
1050                ;
1060                ;
1070                ;                VEDI APPENDICE B1
1080                ;
1090                ;
1100                ;
1110                ;
1120                ;
1130                ;
1140                ;           COPYRIGHT (C) 1983 DI K. SKIER
1150                ;             LEXINGTON, MASSACHUSETTS
1160                ;
1170                ;
1180                ;
1190                ;
1200                ;
1210                ;
1220                ;
1230                ;
1240                ;   NOTA : IL VIC-20 DEVE AVERE L'ESPANSIONE
1250                ;          RAM DA 8K A PARTIRE DALLA LOCAZIONE
1260                ;          $2000. (USARE L'ESPANSIONE VIC-1110
1270                ;          DA 8K OPPURE LA VIC-1111 DA 16K).
1280                ;
1290                ;          IL MONITOR VISIBILE NON E' IN GRADO
1300                ;          DI FUNZIONARE SU UN VIC-20 SENZA
1310                ;          ESPANSIONE O SENZA ALMENO 3K DI
1320                ;          MEMORIA RAM AGGIUNTIVA.
1330                ;
1340                ;
1350                ;
1360                ;
1370                ;
1380                ;
1390                     TV.PTR = $FB        PUNTATORE DELLA POSIZIONE
1400                ;                        SULLO SCHERMO.
1410                ;
1420                     VISMON = $3207      PRIMO LIVELLO DEL
1430                ;                        MONITOR VISIBILE.
1440                ;
1450                ;
1460                     HEX.PG = $3500      PAGINA DI PARTENZA DEL
1470                ;                        DUMP ESADECIMALE.
1480                ;
1490                     MASC   = HEX.PG+$51
1500                     SA     = HEX.PG+$52
1510                     EA     = SA+2
1520                ;
1530                ;
1540                ;              ROUTINE DEL KERNAL:
1550                ;
1560                     CHKOUT = $FFC9
1570                     CHROUT = $FFD2
1580                     CLOSE  = $FFC3
1590                     OPEN   = $FFC0
1600                     SAVE   = $FFD8
1610                     SETLFS = $FFBA
```

```
1620                             SETNAM = $FFBD
1630                    ;
1640                    ;
1650                    ;
1660                    ;
1670                    ;
1680                    ; *******************************************
1690                    ;
1700                    ;               PARAMETRI VIDEO
1710                    ;
1720                    ; *******************************************
1730                    ;
1740                    ;
1750                    ;
1760                    ;
1770                    ;
1780  0000  = 3000             * =  $3000
1790                    ;
1800                    ;
1810                    ;
1820                    ;
1830                    ;
1840  3000  0010        HOME   .WORD $1000    INDIRIZZO DELLA POSIZIONE
1850                    ;                     HOME (ANGOLO SUPERIORE
1860                    ;                     SINISTRO DELLO SCHERMO)
1870                    ;                     IN UN VIC CON ALMENO 8K
1880                    ;                     DI MEMORIA AGGIUNTIVA.
1890  3002  16          RIGINC .BYTE 22       DIFFERENZA TRA GLI INDI-
1900                    ;                     RIZZI DI UNA RIGA CON LA
1905                    ;                     SUCCESSIVA.
1910  3003  15          TVCOL  .BYTE 21       NUMERO DI COLONNE VIDEO
1920                    ;                     CONTANDO DA ZERO.
1930  3004  18          TVRIG  .BYTE 24       NUMERO DI RIGHE VIDEO
1940                    ;                     CONTANDO DA ZERO.
1950  3005  11          HIPAGE .BYTE $11      PAGINA PIU' ALTA DELLA
1955                    ;                     MEMORIA VIDEO.
1960  3006  20          NULL   .BYTE $20      CODICE VIC PER LO SPAZIO
1970                    ;
1980  3007  1E          FRECC  .BYTE $1E      CODICE VIC PER LA FRECCIA
1990                    ;                     VERSO L'ALTO.
2000                    ;
2010                    ;
2020                    ;
2030                    ;
2040                    ;
2050                    ;
2060                    ;
2070                    ;
2080                    ; *******************************************
2090                    ;
2100                    ;                VETTORI DI I/O
2110                    ;
2120                    ; *******************************************
2130                    ;
2140                    ;
2150                    ;
2160                    ;
2170                    ;
2180                    ;
2190  3008  3530        ROMKEY .WORD VICKEY   PUNTATORE CHE INDICA LA
2200                    ;                     ROUTINE CHE LEGGE UN CA-
2210                    ;                     RATTERE ASCII DALLA TA-
2220                    ;                     STIERA
2230                    ;
2240                    ;
2250                    ;
```

```
2260                        ;
2270   300A  3C30           ROMTVT .WORD VICTVT    PUNTATORE CHE INDICA LA
2280                        ;                      ROUTINE CHE STAMPA UN CA-
2290                        ;                      RATTERE ASCII SULLO
2300                        ;                      SCHERMO.
2310   300C  4130           ROMPRT .WORD VICPRT    PUNTATORE CHE INDICA LA
2320                        ;                      ROUTINE DI STAMPA SU
2330                        ;                      STAMPANTE.
2340                        ;
2350   300E  1030           USROUT .WORD COMOD     PUNTATORE CHE INDICA LA
2360                        ;                      ROUTINE UTENTE
2370                        ;
2380                        ;
2390                        ;
2400                        ;
2410                        ;
2420   3010  60             COMOD  RTS             ROUTINE DI COMODO PER
2430                        ;                      ESTENSIONI FUTURE.
2440                        ;
2450                        ;
2460                        ;
2470                        ;
2480                        ;
2490                        ;
2500                        ;  *******************************************
2510                        ;
2520                        ;        CONVERTE L'ASCII NEL CODICE VIDEO
2530                        ;
2540                        ;  *******************************************
2550                        ;
2560                        ;
2570                        ;
2580                        ;
2590                        ;
2600   3011                 FIXCAR                 IL CARATTERE E' IN A.
2610                        ;                      BISOGNA CONVERTIRLO NEL
2620                        ;                      L'APPROPRIATO CODICE
2630                        ;                      VIDEO MA INNANZITUTTO
2640                        ;                      BISOGNA INSERIRE IL CO-
2650                        ;                      DICE COLORE NEL BYTE
2660                        ;                      DELLA MEMORIA COLORE.
2670                        ;
2680                        ;
2690                        ;
2700                        ;
2710   3011  48                    PHA             SALVA IL CARATTERE DA
2720                        ;                      VISUALIZZARE.
2730   3012  A5FC                  LDA TV.PTR+1    SALVA IL BYTE ALTO DI
2740   3014  48                    PHA             TV.PTR.
2750   3015  18                    CLC             TV.PTR PUNTA AL BYTE
2760   3016  6984                  ADC #$84        APPROPRIATO DELLA
2770   3018  85FC                  STA TV.PTR+1    MEMORIA COLORE.
2780   301A  A000                  LDY #0
2790   301C  AD8602                LDA $286        LEGGE IL CODICE DEL CO-
2800                        ;                      LORE CORRENTE E LO SAL-
2810                        ;                      VA NEL BYTE APPROPRIATO
2820                        ;                      DELLA MEMORIA COLORE.
2830   301F  91FB                  STA (TV.PTR),Y
2840                        ;
2850   3021  68                    PLA             RESETTA IL BYTE ALTO DI
2860   3022  85FC                  STA TV.PTR+1    TV.PTR CON IL VALORE
2870                        ;                      ORIGINALE.
2880   3024  68                    PLA             RILEGGE IL CARATTERE DA
2890                        ;                      VISUALIZZARE.
2900                        ;
2910   3025  38                    SEC
```

```
2920  3026  C940            CMP #$40      E' MINORE DI $40?
2930                   ;
2940                   ;
2950  3028  900A            BCC FINFIX    SE SI' NON E' NECESSARIA
2960                   ;                  LA CONVERSIONE.
2970  302A  C960            CMP #$60      E' NELL'INTERVALLO
2980                   ;                  $40 - $5F ?
2990  302C  9003            BCC SUB.40    SE SI' SOTTRAE $40 PER
3000                   ;                  CONVERTIRE DA ASCII A VIC.
3010                   ;
3020                   ;                  E' MAGGIORE DI $5F.
3030
3040  302E  E920            SBC #$20      SOTTRAE $20 PER CONVER-
3050                   ;                  TIRE L'ASCII MINUSCOLO IN
3060                   ;                  CODICE VIC
3070  3030  60              RTS           E RITORNA.
3080                   ;
3090                   ;
3100  3031  38         SUB.40 SEC
3110  3032  E940            SBC #$40      SOTTRAE $40 PER CONVER-
3120                   ;                  TIRE L'ASCII MAIUSCOLO IN
3130  3034  60         FINFIX RTS         CODICE VIC E RITORNA CON
3140                   ;                  IL CODICE VIC DEL CARAT-
3150                   ;                  TERE ORIGINARIAMENTE IN A
3160                   ;
3170                   ;
3180                   ;
3190                   ;
3200                   ;
3210                   ; ********************************************
3220                   ;
3230                   ;    LEGGE IL CARATTERE ASCII DALLA TASTIERA
3240                   ;
3250                   ; ********************************************
3260                   ;
3270                   ;
3280                   ;
3290                   ;
3300  3035  20E4FF     VICKEY JSR $FFE4   LEGGE IL CARATTERE.
3310  3038  AA              TAX           E' ZERO ?
3320  3039  F0FA            BEQ VICKEY    SE SI' NON E' STATO
3330                   ;                  PREMUTO NESSUN TASTO
3340                   ;                  QUINDI RILEGGE.
3350  303B  60              RTS           RITORNA CON IL CARATTERE
3360                   ;                  ASCII DIGITATO.
3370                   ;
3380                   ;
3390                   ;
3400                   ;
3410                   ;
3420                   ; ********************************************
3430                   ;
3440                   ;    VISUALIZZA IL CARATTERE SULLO SCHERMO
3450                   ;
3460                   ; ********************************************
3470                   ;
3480                   ;
3490                   ;
3500                   ;
3510  303C  A201       VICTVT LDX #1      DEFINIAMO IL FILE LOGICO
3520                   ;                  #1 COME CANALE DI OUT.
3530                   ;
3540  303E  4C4330          JMP OUTCAR    INVIA IL CARATTERE IN A
3550                   ;                  AL FILE LOGICO "X"
3560                   ;
3570                   ;
```

```
3580                    ;
3590                    ;
3600                    ;
3610                    ; ******************************************
3620                    ;
3630                    ;    STAMPA IL CARATTERE SULLA STAMPANTE
3640                    ;
3650                    ; ******************************************
3660                    ;
3670                    ;
3680                    ;     QUESTA PROCEDURA PRESUPPONE CHE L'UTENTE
3690                    ;     ABBIA USATO IL BASIC PER APRIRE UNA PE-
3700                    ;     RIFERICA O UN FILE SU DISCO COME FILE
3710                    ;     LOGICO #2 PRIMA DI CHIAMARE IL MONITOR
3720                    ;     VISIBILE.
3730                    ;     IL FILE LOGICO #2 PUO'ESSERE UNA STAMPAN-
3740                    ;     TE, LA PORTA SERIALE RS-232 O PERFINO
3750                    ;     UN FILE SU DISCO O SU CASSETTA.
3760                    ;     IL FATTO IMPORTANTE E' CHE SIA APERTO IN
3770                    ;     MODO DA POTERGLI INVIARE CARATTERI O IL
3780                    ;     TESTO.
3790                    ;
3800                    ;
3810                    ;
3820                    ;
3830  3041  A202        VICPRT LDX #2          DEFINIAMO IL FILE LOGICO
3840                    ;                      #2 COME CANALE DI OUT.
3850                    ;
3860  3043              OUTCAR                 INVIA IL CARATTERE IN A
3870                    ;                      AL FILE LOGICO "X".
3880                    ;
3890                    ;
3900  3043  48                 PHA             SALVA IL CARATTERE DA
3910                    ;                      INVIARE.
3920                    ;
3930  3044  20C9FF             JSR CHKOUT      SELEZIONA IL FILE LOGICO
3940                    ;                      "X"
3950  3047  68                 PLA             RILEGGE IL CARATTERE DA
3960                    ;                      INVIARE.
3970  3048  20D2FF             JSR CHROUT      INVIA IL CARATTERE
3980                    ;
3990                    ;
4000  304B  60                 RTS             E RITORNA.
4010                    ;
4020                    ;
4030                    ;
4040                    ;
4050                    ;
4060                    ; ******************************************
4070                    ;
4080                    ;   MEMORIZZAZIONE DI UN PROGRAMMA IN CODICE
4090                    ;          MACCHINA SU NASTRO O DISCO
4100                    ;
4110                    ; ******************************************
4120                    ;
4130                    ;
4140                    ;
4150                    ;
4160                    ;      E' NECESSARIO SETTARE QUESTE VARIABILI
4170                    ;      PER MEZZO DEL BASIC O DEL MONITOR VISI-
4180                    ;      BILE PRIMA DI RICHIAMARE MLSAVE.
4190                    ;
4200                    ;
4210                    ;
4220                    ;
4230  304C  01          PERIF  .BYTE 1         PERIFERICA DA USARE.
```

```
4240                    ;                     1 SPECIFICA IL DATASSETTE.
4250                    ;                     8 SPECIFICA IL DISK DRIVE.
4260                    ;                     INDIRIZZO DECIMALE 12364.
4270                    ;
4280  304D  00          LENGTH .BYTE 0        LUNGHEZZA NOME FILE.
4290                    ;                     INDIRIZZO DECIMALE 12365.
4300                    ;
4310  304E  00000000    NAME   .BYTE 0,0,0,0  SPAZIO RISERVATO AL NOME
4320  3052  00000000           .BYTE 0,0,0,0  DEL FILE (MAX 20 CARATTERI)
4330  3056  00000000           .BYTE 0,0,0,0
4340  305A  00000000           .BYTE 0,0,0,0
4350  305E  00000000           .BYTE 0,0,0,0
4360                    ;
4370                    ;      NOTA: I PUNTATORI SA E EA DEVONO ESSERE
4380                    ;            SETTATI AGLI INDIRIZZI DI PARTENZA
4390                    ;            E FINE (RISPETTIVAMENTE) DEL PRO-
4400                    ;            GRAMMA DA SALVARE. QUESTO PUO' ES-
4410                    ;            SERE FATTO MOLTO CONVENIENTEMENTE
4420                    ;            CHIAMANDO LA SUBROUTINE SETADS
4430                    ;            ALLA LOCAZIONE $35EA (13795 DEC.)
4440                    ;
4450                    ;
4460                    ;
4470                    ;
4480                    ;
4490  3062  A903        MLSAVE LDA #3         NUMERO DEL FILE LOGICO
4500  3064  AE4C30             LDX PERIF      NUMERO DELLA PERIFERICA
4510  3067  A8                 TAY
4520  3068  20BAFF             JSR SETLFS     CHIAMA LA ROUTINE DEL
4530                    ;                     KERNAL SETLFS. ORA IL VIC
4540                    ;                     SA QUALE PERIFERICA USARE
4550                    ;
4560  306B  AD4D30             LDA LENGTH     LEGGE LA LUNGHEZZA DEL
4570                    ;                     NOME FILE.
4580  306E  A24E               LDX #LOW(NAME)
4590  3070  A030               LDY #HIGH(NAME)
4600                    ;                     (X,Y) PUNTA AL NOME
4610                    ;                     FILE.
4620  3072  20BDFF             JSR SETNAM     ROUTINE KERNAL "SETNAM".
4630                    ;                     ORA IL VIC SA IL NOME DEL
4640                    ;                     FILE DA CREARE.
4650                    ;
4660  3075  AD5235             LDA SA
4670  3078  85FD               STA $FD
4680  307A  AD5335             LDA SA+1
4690  307D  85FE               STA $FE        IL PUNTATORE A $FD INDICA
4700                    ;                     L'INDIRIZZO DEL PROGRAMMA
4710  307F  A9FD               LDA #$FD       A CONTIENE L'OFFSET DI
4720                    ;                     QUEL PUNTATORE.
4730  3081  AE5435             LDX EA
4740  3084  AC5535             LDY EA+1       ORA (X,Y) PUNTA ALLA FINE
4750                    ;                     DEL PROGRAMMA.
4760                    ;                     MA LA ROUTINE KERNAL SAVE
4770                    ;                     RICHIEDE CHE (X,Y) PUNTI
4780                    ;                     UN BYTE PIU' IN ALTO DELLA
4790                    ;                     FINE QUINDI INCREMENTA
4800                    ;                     (X,Y)
4810  3087  E8                 INX
4820  3088  D001               BNE XY.SET
4830  308A  C8                 INY
4840                    ;
4850                    ;
4860  308B  20D8FF      XY.SET JSR SAVE       ROUTINE KERNAL SAVE
4870                    ;                     APRE IL FILE E MEMORIZZA
4880                    ;                     LA PORZIONE DI MEMORIA
4890                    ;                     SPECIFICATA.
```

```
4900                        ;
4910                        ;
4920   308E  60                  RTS           RITORNA (PRESUMIBILMENTE
4930                        ;                  AL BASIC O AL MONITOR).
4940                        ;
4950                        ;
4960                        ;
4970                        ;
4980                        ;
4990                        ; ******************************************
5000                        ;
5010                        ;     MONITOR VISIBILE: CHIAMATE DA BASIC
5020                        ;
5030                        ; ******************************************
5040                        ;
5050                        ;
5060                        ;
5070                        ;
5080   308F  A900          ENTRY LDA #0
5090   3091  48                  PHA
5100   3092  28                  PLP           ORA IL REGISTRO P = 0
5110
5120                        ;
5130                        ;                  ATTIVA LO SCHERMO COME
5140                        ;                  FILE LOGICO #1
5150                        ;
5160   3093  20BDFF              JSR SETNAM    A E' GIA' A $00, PER
5170                        ;                  INDICARE NESSUN NOME FILE
5180                        ;
5190   3096  A901                LDA #1        NUMERO DEL FILE LOGICO.
5200   3098  A200                LDX #0        NUMERO DELLA PERIFERICA
5210   309A  A0FF                LDY #255      SCHERMO.
5220   309C  20BAFF              JSR SETLFS    ROUTINE KERNAL SETLFS
5230                        ;
5240   309F  20C0FF              JSR OPEN      ORA LO SCHERMO E' IL FILE
5250                        ;                  LOGICO #1.
5260                        ;
5270                        ;
5280   30A2  A903                LDA #3        SETTA LA MASCHERA A 3.
5290   30A4  8D5135              STA MASC      TVDUMP VISUALIZZERA' OGNI
5300                        ;                  LINEA ESADECIMALE SU 4
5310                        ;                  LINEE DI SCHERMO.
5320                        ;
5330   30A7  200732              JSR VISMON    CHIAMA IL MONITOR VISIBILE
5340                        ;
5350                        ;
5360                        ;
5370                        ;
5380                        ;
5390                        ;
5400   30AA  A901                LDA #1
5410   30AC  20C3FF              JSR CLOSE
5420                        ;
5430   30AF  60                  RTS           RITORNA (PRESUMIBILMENTE
                            ;                  AL BASIC).
```

CROSS REFERENCE

```
FRECC  3007     NULL   3006     CHKOUT FFC9     CHROUT FFD2
CLOSE  FFC3     PERIF  304C     COMOD  3010     EA     3554
ENTRY  308F     FIXCAR 3011     FINFIX 3034     HEX.PG 3500
```

```
HIPAGE 3005      HOME   3000      LENGTH 304D      MASC   3551
MLSAVE 3062      NAME   304E      OPEN   FFC0      OUTCAR 3043
ROMKEY 3008      ROMPRT 300C      ROMTVT 300A      RIGINC 3002
SA     3552      SAVE   FFD8      SETLFS FFBA      SETNAM FFBD
SUB.40 3031      TV.PTR 00FB      TVCOL  3003      TVRIG  3004
USROUT 300E      VICKEY 3035      VICPRT 3041      VICTVT 303C
VISMON 3207      XY.SET 308B
```

C.14 Dati di sistema per il C-64

```
1000                  ;        APPENDICE C14: LISTATO ASSEMBLER DEL
1010                  ;             BLOCCO DEI DATI DI SISTEMA
1020                  ;                PER IL COMMODORE 64
1030                  ;
1040                  ;
1050                  ;
1060                  ;
1070                  ;                VEDI APPENDICE B2
1080                  ;
1090                  ;
1100                  ;
1110                  ;
1120                  ;
1130                  ;
1140                  ;          COPYRIGHT (C) 1983 DI K. SKIER
1150                  ;             LEXINGTON, MASSACHUSETTS
1160                  ;
1170                  ;
1180                  ;
1190                  ;
1200                  ;
1210                  ;
1220                  ;
1230                  ;
1240                  ;
1250                  ;
1260                    TV.PTR = $FB       PUNTATORE DELLA POSI-
1270                  ;                    ZIONE SULLO SCHERMO.
1280                  ;
1290                  ;
1300                    VISMON = $3207     PRIMO LIVELLO DEL
1310                  ;                    MONITOR VISIBILE.
1320                  ;
1330                  ;
1340                    HEX.PG = $3500     PAGINA DI PARTENZA DEL
1350                  ;                    DUMP ESADECIMALE.
1360                  ;
1370                    SA     = HEX.PG+$52
1380                    EA     = SA+2
1390                  ;
1400                  ;
1410                  ;              ROUTINE DEL KERNAL:
1420                  ;
1430                    CHKOUT = $FFC9
1440                    CHROUT = $FFD2
1450                    CLOSE  = $FFC3
1460                    OPEN   = $FFC0
1470                    SAVE   = $FFD8
1480                    SETLFS = $FFBA
1490                    SETNAM = $FFBD
1500                  ;
1510                  ;
1520                  ;
1530                  ;
1540                  ;
1550                  ; *****************************************
1560                  ;
1570                  ;             PARAMETRI VIDEO
1580                  ;
1590                  ; *****************************************
1600                  ;
1610                  ;
```

```
1620                            ;
1630                            ;
1640                            ;
1650    0000   = 3000                 * =  $3000
1660                            ;
1670                            ;
1680                            ;
1690                            ;
1700                            ;
1710    3000   0004             HOME   .WORD $400     INDIRIZZO DELLA POSIZIO-
1720                            ;                     NE HOME (ANGOLO SUPERIORE
1730                            ;                     SINISTRO DELLO SCHERMO)
1740    3002   28               RIGINC .BYTE $28      DIFFERENZA TRA GLI INDI-
1750                            ;                     RIZZI DI UNA RIGA CON LA
1755                            ;                     SUCCESSIVA.
1760    3003   27               TVCOL  .BYTE 39       NUMERO DI COLONNE VIDEO
1770                            ;                     CONTANDO DA ZERO.
1780    3004   18               TVRIG  .BYTE 24       NUMERO DI RIGHE VIDEO
1790                            ;                     CONTANDO DA ZERO.
1800    3005   07               HIPAGE .BYTE $07      PAGINA PIU' ALTA DELLA
1805                            ;                     MEMORIA VIDEO.
1810    3006   20               NULL   .BYTE $20      CODICE C64 PER LO SPAZIO.
1820                            ;
1830    3007   1E               FRECC  .BYTE $1E      CODICE C64 PER LA FRECCIA
1840                            ;                     VERSO L'ALTO.
1850                            ;
1860                            ;
1870                            ;
1880                            ;
1890                            ;
1900                            ;
1910                            ;
1920                            ;
1930                            ; ********************************************
1940                            ;
1950                            ;              VETTORI DI I/O
1960                            ;
1970                            ; ********************************************
1980                            ;
1990                            ;
2000                            ;
2010                            ;
2020                            ;
2030                            ;
2040    3008   3530             ROMKEY .WORD C64KEY   PUNTATORE CHE INDICA LA
2050                            ;                     ROUTINE CHE LEGGE UN CA-
2060                            ;                     RATTERE ASCII DALLA TA-
2070                            ;                     STIERA.
2080                            ;
2090                            ;
2100                            ;
2110                            ;
2120    300A   3C30             ROMTVT .WORD C64TVT   PUNTATORE CHE INDICA LA
2130                            ;                     ROUTINE CHE STAMPA UN CA-
2140                            ;                     RATTERE ASCII SULLO
2150                            ;                     SCHERMO.
2160    300C   4130             ROMPRT .WORD C64PRT   PUNTATORE CHE INDICA LA
2170                            ;                     ROUTINE DI STAMPA SU
2180                            ;                     STAMPANTE.
2190                            ;
2200    300E   1030             USROUT .WORD COMOD    PUNTATORE CHE INDICA LA
2210                            ;                     ROUTINE UTENTE.
2220                            ;
2230                            ;
2240                            ;
2250                            ;
```

```
2260                    ;
2270  3010  60          COMOD   RTS              ROUTINE DI COMODO PER
2280                    ;                        ESTENSIONI FUTURE.
2290                    ;
2300                    ;
2310                    ;
2320                    ;
2330                    ;
2340                    ;
2350                    ; ****************************************
2360                    ;
2370                    ;     CONVERTE L' ASCII NEL CODICE VIDEO
2380                    ;
2390                    ; ****************************************
2400                    ;
2410                    ;
2420                    ;
2430                    ;
2440                    ;
2450  3011              FIXCAR                   IL CARATTERE E' IN A.
2460                    ;                        BISOGNA CONVERTIRLO NEL-
2470                    ;                        L'APPROPRIATO CODICE
2480                    ;                        VIDEO, MA INNANZITUTTO
2490                    ;                        BISOGNA INSERIRE IL CODI-
2500                    ;                        CE COLORE NEL BYTE DELLA
2510                    ;                        MEMORIA COLORE.
2520                    ;
2530                    ;
2540                    ;
2550                    ;
2560  3011  48                  PHA              SALVA IL CARATTERE DA
2570                    ;                        VISUALIZZARE.
2580  3012  A5FC                LDA TV.PTR+1     SALVA IL BYTE ALTO DI
2590  3014  48                  PHA              TV.PTR.
2600  3015  18                  CLC              TV.PTR PUNTA AL BYTE
2610  3016  69D4                ADC #$D4         APPROPRIATO DELLA
2620  3018  85FC                STA TV.PTR+1     MEMORIA COLORE.
2630  301A  A000                LDY #0
2640  301C  AD8602              LDA $286         LEGGE IL CODICE DEL CO-
2650                    ;                        LORE CORRENTE E LO SALVA
2660                    ;                        NEL BYTE APPROPRIATO DEL-
2670                    ;                        LA MEMORIA COLORE.
2680  301F  91FB                STA (TV.PTR),Y
2690                    ;
2700  3021  68                  PLA              RESETTA IL BYTE ALTO DI
2710  3022  85FC                STA TV.PTR+1     TV.PTR  CON IL VALORE
2720                    ;                        ORIGINALE.
2730  3024  68                  PLA              RILEGGE IL CARATTERE DA
2740                    ;                        VISUALIZZARE.
2750                    ;
2760  3025  38                  SEC
2770  3026  C940                CMP #$40         E' MINORE DI $40?
2780                    ;
2790                    ;
2800  3028  900A                BCC FINFIX       SE SI',NON E' NECESSARIA
2810                    ;                        LA CONVERSIONE.
2820  302A  C960                CMP #$60         E' NELL'INTERVALLO
2830                    ;                        $40 - $5F?
2840  302C  9003                BCC SUB.40       SE SI' SOTTRAE $40 PER
2850                    ;                        CONVERTIRE DA ASCII A C64.
2860                    ;
2870                    ;                        E' MAGGIORE DI $5F.
2880
2890  302E  E920                SBC #$20         SOTTRAE $20 PER CONVERTI-
2900                    ;                        RE L'ASCII MINUSCOLO IN
2910                    ;                        CODICE C64
```

```
2920   3030  60                  RTS             E RITORNA.
2930                     ;
2940                     ;
2950   3031  38          SUB.40  SEC
2960   3032  E940                SBC #$40        SOTTRAE $40 PER CONVERTI-
2970                     ;                       RE L'ASCII MAIUSCOLO IN
2980   3034  60          FINFIX  RTS             CODICE C64 E RITORNA CON
2990                     ;                       IL CODICE C64  DEL CARAT-
3000                     ;                       TERE ORIGINARIAMENTE IN A
3010                     ;
3020                     ;
3030                     ;
3040                     ;
3050                     ;
3060                     ; ********************************************
3070                     ;
3080                     ;    LEGGE IL CARATTERE ASCII DALLA TASTIERA
3090                     ;
3100                     ; ********************************************
3110                     ;
3120                     ;
3130                     ;
3140                     ;
3150   3035  20E4FF      C64KEY  JSR $FFE4       LEGGE IL CARATTERE.
3160   3038  AA                  TAX             E' ZERO?
3170   3039  F0FA                BEQ C64KEY      SE SI' NON E' STATO PRE-
3180                     ;                       MUTO NESSUN TASTO, QUIN-
3190                     ;                       DI RILEGGE.
3200   303B  60                  RTS             RITORNA CON IL CARATTERE
3210                     ;                       ASCII DIGITATO.
3220                     ;
3230                     ;
3240                     ;
3250                     ;
3260                     ;
3270                     ; ********************************************
3280                     ;
3290                     ;    VISUALIZZA IL CARATTERE SULLO SCHERMO
3300                     ;
3310                     ; ********************************************
3320                     ;
3330                     ;
3340                     ;
3350                     ;
3360   303C  A201        C64TVT  LDX #1          DEFINIAMO IL FILE LOGICO
3370                     ;                       #1 COME CANALE DI OUT.
3380                     ;
3390   303E  4C4330              JMP OUTCAR      INVIA IL CARATTERE IN A
3400                     ;                       AL FILE LOGICO "X"
3410                     ;
3420                     ;
3430                     ;
3440                     ;
3450                     ;
3460                     ; ********************************************
3470                     ;
3480                     ;      STAMPA IL CARATTERE SULLA STAMPANTE
3490                     ;
3500                     ; ********************************************
3510                     ;
3520                     ;
3530                     ;   QUESTA PROCEDURA PRESUPPONE CHE L'UTENTE
3540                     ;   ABBIA USATO IL BASIC PER APRIRE UNA PERI-
3550                     ;   FERICA O UN FILE SU DISCO COME FILE LOGI-
3560                     ;   CO #2 PRIMA DI CHIAMARE IL MONITOR VISI-
3570                     ;   BILE.
```

```
3580                      ;   IL FILE LOGICO #2 PUO' ESSERE UNA STAMPAN-
3590                      ;   TE, LA PORTA SERIALE RS-232 O PERFINO UN
3600                      ;   FILE SU DISCO O SU CASSETTA.
3610                      ;   IL FATTO IMPORTANTE E' CHE SIA APERTO IN
3620                      ;   MODO DA POTERGLI INVIARE CARATTERI O IL
3630                      ;   TESTO.
3640                      ;
3650                      ;
3660                      ;
3670                      ;
3680  3041  A202          C64PRT LDX #2          DEFINIAMO IL FILE LOGICO
3690                      ;                      #2 COME CANALE DI OUT.
3700                      ;
3710  3043                OUTCAR                 INVIA IL CARATTERE IN A
3720                      ;                      AL FILE LOGICO "X".
3730                      ;
3740                      ;
3750  3043  48                   PHA             SALVA IL CARATTERE DA
3760                      ;                      INVIARE.
3770                      ;
3780  3044  20C9FF               JSR CHKOUT      SELEZIONA IL FILE LOGICO
3790                      ;                      "X".
3800  3047  68                   PLA             RILEGGE IL CARATTERE DA
3810                      ;                      INVIARE.
3820  3048  20D2FF               JSR CHROUT      INVIA IL CARATTERE
3830                      ;
3840                      ;
3850  304B  60                   RTS             E RITORNA.
3860                      ;
3870                      ;
3880                      ;
3890                      ;
3900                      ;
3910                      ; *******************************************
3920                      ;
3930                      ;   MEMORIZZAZIONE DI UN PROGRAMMA IN CODICE
3940                      ;          MACCHINA SU NASTRO O DISCO
3950                      ;
3960                      ; *******************************************
3970                      ;
3980                      ;
3990                      ;
4000                      ;
4010                      ;      E' NECESSARIO SETTARE QUESTE VARIABILI
4020                      ;      PER MEZZO DEL BASIC O DEL MONITOR VISI-
4030                      ;      BILE PRIMA DI RICHIAMARE MLSAVE
4040                      ;
4050                      ;
4060                      ;
4070                      ;
4080  304C  01            PERIF  .BYTE 1         PERIFERICA DA USARE.
4090                      ;                      1 SPECIFICA IL DATASSETTE.
4100                      ;                      8 SPECIFICA IL DISK DRIVE.
4110                      ;                      INDIRIZZO DECIMALE 12364.
4120                      ;
4130  304D  00            LENGTH .BYTE 0         LUNGHEZZA NOMEFILE.
4140                      ;                      INDIRIZZO DECIMALE 12365.
4150                      ;
4160  304E  00000000      NAME   .BYTE 0,0,0,0   SPAZIO RISERVATO AL NOME
4170  3052  00000000             .BYTE 0,0,0,0   DEL FILE (MAX 20 CARATTERI)
4180  3056  00000000             .BYTE 0,0,0,0
4190  305A  00000000             .BYTE 0,0,0,0
4200  305E  00000000             .BYTE 0,0,0,0
4210                      ;
4220                      ;----------------->    NOTA: I PUNTATORI SA E EA
4230                      ;                      DEVONO ESSERE SETTATI A-
```

```
4240                         ;                  GLI INDIRIZZI DI PARTENZA
4250                         ;                  E FINE (RISPETTIVAMENTE)
4260                         ;                  DEL PROGRAMMA DA SALVARE.
4270                         ;                  QUESTO PUO' ESSERE CONVE-
4280                         ;                  NIENTE CHIAMANDO LA SUB-
4290                         ;                  ROUTINE SETADS ALLA LOCA-
4300                         ;                  ZIONE $35EA (13795 DEC.).
4310                         ;
4320                         ;
4330                         ;
4340  3062  A903       MLSAVE LDA #3            NUMERO DEL FILE LOGICO.
4350  3064  AE4C30            LDX PERIF         NUMERO DELLA PERIFERICA
4360  3067  A8                TAY
4370  3068  20BAFF            JSR SETLFS        CHIAMA LA ROUTINE DEL
4380                         ;                  KERNAL SETLFS. ORA IL C64
4390                         ;                  SA QUALE PERIFERICA USARE
4400                         ;
4410  306B  AD4D30            LDA LENGTH        LEGGE LA LUNGHEZZA DEL
4420                         ;                  NOME FILE.
4430  306E  A24E              LDX #LOW(NAME)
4440  3070  A030              LDY #HIGH(NAME)
4450                         ;                  (X,Y) PUNTA AL NOME FILE
4460                         ;
4470  3072  20BDFF            JSR SETNAM        ROUTINE KERNAL "SETNAM".
4480                         ;                  ORA IL C64 SA IL NOME DEL
4490                         ;                  FILE DA CREARE.
4500                         ;
4510  3075  AD5235            LDA SA
4520  3078  85FD              STA $FD
4530  307A  AD5335            LDA SA+1
4540  307D  85FE              STA $FE           IL PUNTATORE A $FD INDICA
4550                         ;                  L'INDIRIZZO DEL PROGRAMMA
4560  307F  A9FD              LDA #$FD          A CONTIENE L'OFFSET DI
4570                         ;                  QUEL PUNTATORE.
4580  3081  AE5435            LDX EA
4590  3084  AC5535            LDY EA+1          ORA (X,Y) PUNTA ALLA FINE
4600                         ;                  DEL PROGRAMMA.
4610                         ;                  MA LA ROUTINE KERNAL SAVE
4620                         ;                  RICHIEDE CHE (X,Y) PUNTI
4630                         ;                  UN BYTE PIU' IN ALTO DEL-
4640                         ;                  LA FINE QUINDI INCREMENTA
4650                         ;                  (X,Y)
4660  3087  E8                INX
4670  3088  D001              BNE XY.SET
4680  308A  C8                INY
4690                         ;                  ORA (X,Y) E' SETTATO.
4700                         ;
4710  308B  20D8FF     XY.SET JSR SAVE          ROUTINE KERNAL SAVE
4720                         ;                  APRE IL FILE E MEMORIZZA
4730                         ;                  LA PORZIONE DI MEMORIA
4740                         ;                  SPECIFICATA.
4750                         ;
4760                         ;
4770  308E  60                RTS               RITORNA (PRESUMIBILMENTE
4780                         ;                  AL BASIC O AL MONITOR VI-
4790                         ;                  SIBILE).
4800                         ;
4810                         ;
4820                         ;
4830                         ;
4840                         ; *******************************************
4850                         ;
4860                         ;    MONITOR VISIBILE: CHIAMATE DA BASIC
4870                         ;
4880                         ; *******************************************
4890                         ;
```

```
4900                      ;
4910                      ;
4920                      ;
4930  308F  A900          ENTRY   LDA #0
4940  3091  48                    PHA
4950  3092  28                    PLP          ORA IL REGISTRO P = 0
4960
4970                      ;
4980                      ;                    ATTIVA LO SCHERMO COME
4990                      ;                    FILE LOGICO #1
5000                      ;
5010  3093  20BDFF               JSR SETNAM    A E' GIA' A $00, PER
5020                      ;                    INDICARE NESSUN NOME FILE
5030                      ;
5040  3096  A901                 LDA #1        NUMERO DEL FILE LOGICO.
5050  3098  A200                 LDX #0        NUMERO DELLA PERIFERICA
5060  309A  A0FF                 LDY #255      SCHERMO.
5070  309C  20BAFF               JSR SETLFS    ROUTINE KERNAL SETLFS
5080                      ;
5090  309F  20C0FF               JSR OPEN      ORA LO SCHERMO E' IL FILE
5100                      ;                    LOGICO #1.
5110                      ;
5120  30A2  200732               JSR VISMON    CHIAMA IL MONITOR VISIBILE
5130                      ;
5140                      ;
5150                      ;
5160                      ;
5170                      ;
5180                      ;
5190  30A5  A901                 LDA #1
5200  30A7  20C3FF               JSR CLOSE
5210                      ;
5220  30AA  60                   RTS           RITORNA (PRESUMIBILMENTE
5230                      ;                    AL BASIC).
```

CROSS REFERENCE

```
FRECC  3007
C64TVT 303C   NULL   3006   C64KEY 3035   C64PRT 3041
PERIF  304C   CHKOUT FFC9   CHROUT FFD2   CLOSE  FFC3
FIXCAR 3011   COMOD  3010   EA     3554   ENTRY  308F
HOME   3000   FINFIX 3034   HEX.PG 3500   HIPAGE 3005
OPEN   FFC0   LENGTH 304D   MLSAVE 3062   NAME   304E
ROMTVT 300A   OUTCAR 3043   ROMKEY 3008   ROMPRT 300C
SETLFS FFBA   RIGINC 3002   SA     3552   SAVE   FFD8
TVCOL  3003   SETNAM FFBD   SUB.40 3031   TV.PTR 00FB
XY.SET 308B   TVRIG  3004   USROUT 300E   VISMON 3207
```

Appendice

D Dump esadecimali

D.1 Utility per la gestione dello schermo

```
APPENDICE D1:          UTILITY DELLO SCHERMO


VEDI CAPITOLO 5


DUMP  $3100-$31E1

        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3100    20 C4 31 20 2B 31 AE 03 30 AC 04 30 20 13 31 20
3110    D3 31 60 8E 2A 31 98 AA AD 06 30 AC 2A 31 91 FB
3120    88 10 FB 20 76 31 CA 10 EF 60 27 A2 00 A0 00 18
3130    90 0A AD 04 30 4A A8 AD 03 30 4A AA 38 EC 03 30
3140    90 03 AE 03 30 38 CC 04 30 90 03 AC 04 30 AD 00
3150    30 85 FB AD 01 30 85 FC 08 D8 8A 18 65 FB 90 03
3160    E6 FC 18 C0 00 F0 0B 18 6D 02 30 90 02 E6 FC 88
3170    D0 F5 85 FB 28 60 AD 02 30 18 90 05 20 9B 31 A9
3180    01 08 D8 18 65 FB 90 02 E6 FC 85 FB 38 AD 05 30
3190    C5 FC B0 05 AD 01 30 85 FC 28 60 20 11 30 A0 00
31A0    91 FB 60 48 4A 4A 4A 4A 20 B6 31 20 7C 31 68 20
31B0    B6 31 20 7C 31 60 08 D8 29 0F C9 0A 30 02 69 06
31C0    69 30 28 60 68 AA 68 A8 A5 FC 48 A5 FB 48 98 48
31D0    8A 48 60 68 AA 68 A8 68 85 FB 68 85 FC 98 48 8A
31E0    48 60

----------> FINE  APPENDICE D1  <----------
```

D.2 Il Monitor Visibile
(primo livello e subroutine di utility)

```
APPENDICE D2:             IL MONITOR VISIBILE
                   (PRIMO LIVELLO E SUBROUTINE VIDEO)


VEDI CAPITOLO 6


DUMP $3200-$32D3

        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3200    00 00 00 00 00 05 32 08 D8 20 12 32 20 E3 32 18
3210    90 F6 20 C4 31 20 25 32 20 35 32 20 5D 32 20 B0
3220    32 20 D3 31 60 A2 00 A0 00 20 3C 31 AE 03 30 A0
3230    03 20 13 31 60 A2 0B A0 00 20 3C 31 A0 00 8C 52
3240    32 B9 53 32 20 7C 31 EE 52 32 AC 52 32 C0 0A D0
3250    F0 60 0A 41 20 20 58 20 20 59 20 20 50 A2 00 A0
3260    01 20 3C 31 AD 06 32 20 A3 31 AD 05 32 20 A3 31
3270    20 7F 31 20 95 32 48 20 A3 31 20 7F 31 68 20 7C
3280    31 20 7F 31 A2 00 BD 01 32 20 A3 31 20 7F 31 E8
3290    E0 04 D0 F2 60 A5 FB 48 A6 FC AD 05 32 85 FB AD
32A0    06 32 85 FC A0 00 B1 FB A8 68 85 FB 86 FC 98 60
32B0    AC 00 32 38 C0 07 90 05 A0 00 8C 00 32 B9 CD 32
32C0    AA A0 02 20 3C 31 AD 07 30 20 7C 31 60 03 06 08
32D0    0B 0E 11 14

----------> FINE APPENDICE D2  <----------
```

D.3 Il Monitor Visibile
(subroutine di aggiornamento)

```
APPENDICE D3:                    MONITOR VISIBILE
                                (SUBROUTINE AGGIOR)


 VEDI CAPITOLO 6


DUMP $32E0-$33EE


        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

32E0    6C 08 30 20 E0 32 C9 1D D0 10 EE 00 32 AD 00 32
32F0    C9 07 D0 05 A9 00 8D 00 32 60 C9 9D D0 0B CE 00
3300    32 10 05 A9 06 8D 00 32 60 C9 20 D0 09 EE 05 32
3310    D0 03 EE 06 32 60 C9 0D D0 0C AD 05 32 D0 03 CE
3320    06 32 CE 05 32 60 AE 00 32 E0 02 D0 1B A8 A5 FB
3330    48 A6 FC AD 05 32 85 FB AD 06 32 85 FC 98 A0 00
3340    91 FB 86 FC 68 85 FB 60 C9 47 D0 23 AC 03 32 AE
3350    02 32 AD 04 32 48 AD 01 32 28 20 6C 33 08 8D 01
3360    32 8E 02 32 8C 03 32 68 8D 04 32 60 6C 05 32 48
3370    20 D5 33 30 4B A8 68 98 AE 00 32 D0 14 A2 03 18
3380    0E 05 32 2E 06 32 CA 10 F6 98 0D 05 32 8D 05 32
3390    60 E0 01 D0 18 29 0F 48 20 95 32 0A 0A 0A 0A 29
33A0    F0 8D AC 33 68 0D AC 33 20 2D 33 60 00 CA CA CA
33B0    A0 03 18 1E 01 32 88 10 F9 1D 01 32 9D 01 32 60
33C0    68 C9 93 D0 04 20 00 31 60 C9 51 D0 04 68 68 28
33D0    60 20 10 30 60 38 E9 30 90 0F C9 0A 90 0E E9 07
33E0    C9 10 B0 05 38 C9 0A B0 03 A9 FF 60 A2 00 60

---------->  FINE APPENDICE D3  <----------
```

D.4 Utility di stampa

```
APPENDICE D4:            UTILITY DI STAMPA


 VEDI CAPITOLO 7


DUMP $3400-$3543

        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3400    FF FF 00 20 00 00 0C 35 A9 FF 8D 01 34 60 A9 00
3410    8D 01 34 60 A9 FF 8D 00 34 60 A9 00 8D 00 34 60
3420    A9 FF 8D 02 34 60 A9 00 8D 02 34 60 20 08 34 20
3430    14 34 20 20 34 60 20 0E 34 20 1A 34 20 26 34 60
3440    C9 00 F0 24 8D 03 34 AD 01 34 F0 06 AD 03 34 20
3450    69 34 AD 00 34 F0 06 AD 03 34 20 6C 34 AD 02 34
3460    F0 06 AD 03 34 20 6F 34 60 6C 0A 30 6C 0C 30 6C
3470    0E 30 A9 0D 20 40 34 A9 0A 20 40 34 60 A9 20 20
3480    40 34 60 48 4A 4A 4A 4A 20 B6 31 20 40 34 68 20
3490    B6 31 20 40 34 60 A9 20 8E 04 34 48 AE 04 34 F0
34A0    0A CE 04 34 20 40 34 68 18 90 F0 68 60 8E 04 34
34B0    AE 04 34 F0 09 CE 04 34 20 72 34 18 90 F2 60 8E
34C0    05 34 B5 01 48 B5 00 48 AE 05 34 A1 00 C9 FF F0
34D0    0C F6 00 D0 02 F6 01 20 40 34 18 90 EB 68 95 00
34E0    68 95 01 60 68 AA 68 A8 20 12 35 8E 05 32 8C 06
34F0    32 20 0D 33 20 0D 33 20 95 32 C9 FF F0 06 20 40
3500    34 18 90 F0 AE 05 32 AC 06 32 20 2B 35 98 48 8A
3510    48 60 68 8D 06 34 68 8D 07 34 AD 06 32 48 AD 05
3520    32 48 AD 07 34 48 AD 06 34 48 60 68 8D 06 34 68
3530    8D 07 34 68 8D 05 32 68 8D 06 32 AD 07 34 48 AD
3540    06 34 48 60

----------> FINE APPENDICE D4 <----------
```

D.5 Il dump esadecimale

```
APPENDICE D5:              DUMP ESADECIMALE


 VEDI CAPITOLO 8

DUMP $3550-$37A6

         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3550    00 07 50 35 A6 37 00 20 08 34 A9 04 8D 50 35 AD
3560    05 32 29 F0 8D 05 32 20 9B 35 20 7D 34 20 7D 34
3570    20 94 35 20 0D 33 AD 05 32 2D 51 35 D0 EF 20 72
3580    34 AD 05 32 29 0F D0 03 20 72 34 CE 50 35 D0 D7
3590    20 0E 34 60 20 95 32 20 83 34 60 AD 06 32 20 83
35A0    34 AD 05 32 20 83 34 60 20 C3 35 20 E3 35 20 9A
35B0    37 20 14 34 20 E5 36 20 3C 37 10 FB 20 72 34 20
35C0    1A 34 60 20 08 34 20 1A 34 20 E4 34 7F 0D 50 52
35D0    49 4E 54 49 4E 47 20 48 45 58 44 55 4D 50 0D 0A
35E0    0A FF 60 20 08 34 20 E4 34 7F 0D 0A 53 45 54 20
35F0    53 54 41 52 54 49 4E 47 20 41 44 44 52 45 53 53
3600    20 41 4E 44 20 50 52 45 53 53 20 22 51 22 2E FF
3610    20 07 32 20 61 36 20 08 34 20 E4 34 7F 0D 0A 53
3620    45 54 20 45 4E 44 20 41 44 44 52 45 53 53 20 41
3630    4E 44 20 50 52 45 53 53 20 22 51 22 2E FF 20 07
3640    32 38 AD 06 32 CD 53 35 90 24 D0 08 AD 05 32 CD
3650    52 35 90 1A AD 06 32 8D 55 35 AD 05 32 8D 54 35
3660    60 AD 06 32 8D 53 35 AD 05 32 8D 52 35 60 20 E4
3670    34 7F 0D 0A 0A 0A 20 45 52 52 4F 52 21 21 21 20
3680    45 4E 44 20 41 44 44 52 45 53 53 20 4C 45 53 53
3690    20 54 48 41 4E 20 53 54 41 52 54 20 41 44 44 52
36A0    45 53 53 2C 20 57 48 49 43 48 20 49 53 20 FF 20
36B0    B5 36 4C 16 36 A9 24 20 40 34 AD 53 35 20 83 34
36C0    AD 52 35 20 83 34 60 A9 24 20 40 34 AD 55 35 20
36D0    83 34 AD 54 35 20 83 34 60 20 B5 36 A9 2D 20 40
36E0    34 20 C7 36 60 20 E4 34 7F 0D 0A 0A 44 55 4D 50
36F0    49 4E 47 20 FF 20 D9 36 20 72 34 20 E4 34 7F 0A
3700    0A 20 20 20 20 20 20 20 20 30 20 20 31 20 20 32
3710    20 20 33 20 20 34 20 20 35 20 20 36 20 20 37 20
3720    20 38 20 20 39 20 20 41 20 20 42 20 20 43 20 20
3730    44 20 20 45 20 20 46 0D 0A 0A FF 60 20 72 34 AD
3740    05 32 48 29 0F 8D 56 35 68 29 F0 8D 05 32 20 9B
3750    35 A2 03 20 96 34 AD 56 35 F0 0D A2 03 20 96 34
3760    20 0D 33 CE 56 35 D0 F3 20 94 35 20 7D 34 20 7D
3770    37 30 09 AD 05 32 29 0F C9 00 D0 EC 60 38 AD 06
3780    32 CD 55 35 90 0B D0 0F 38 AD 05 32 CD 54 35 B0
3790    06 20 0D 33 A9 00 60 A9 FF 60 AD 52 35 8D 05 32
37A0    AD 53 35 8D 06 32 60

-----------> FINE APPENDICE D5 <-----------
```

D.6 Un disassemblatore
(primo livello e subroutine di utility)

APPENDICE D6: DISASSEMBLER (PRIMO LIVELLO E UTILITY)

VEDI CAPITOLO 9

DUMP $3900-$3A42

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3900	05	00	00	00	00	00	00	00	0B	20	08	34	AD	00	39	8D
3910	01	39	A9	FF	8D	54	35	8D	55	35	20	72	34	20	7D	39
3920	CE	01	39	D0	F8	60	20	1A	34	20	08	34	20	E4	34	7F
3930	0D	0A	20	20	20	20	20	50	52	49	4E	54	49	4E	47	20
3940	44	49	53	41	53	53	45	4D	42	4C	45	52	2E	0D	0A	FF
3950	20	E3	35	20	14	34	20	E4	34	7F	0D	0A	44	49	53	41
3960	53	53	45	4D	42	4C	49	4E	47	20	FF	20	D9	36	20	9A
3970	37	20	72	34	20	7D	39	10	FB	20	1A	34	60	20	95	32
3980	48	20	92	39	20	7D	34	68	20	AF	39	20	01	3A	20	7D
3990	37	60	A2	03	8E	02	39	AA	BD	00	3C	AA	BD	50	3B	8E
39A0	03	39	20	40	34	AE	03	39	E8	CE	02	39	D0	EE	60	AA
39B0	BD	00	3D	AA	20	B8	39	60	BD	26	3B	8D	04	39	E8	BD
39C0	26	3B	8D	05	39	6C	04	39	20	0D	33	20	94	35	60	20
39D0	0D	33	20	95	32	48	20	0D	33	20	94	35	68	20	83	34
39E0	60	A9	28	D0	02	A9	29	20	40	34	60	A9	2C	20	40	34
39F0	A9	58	20	40	34	60	A9	2C	20	40	34	A9	59	20	40	34
3A00	60	8D	07	39	8E	06	39	CA	30	06	20	1A	33	CA	10	FA
3A10	08	D8	38	AD	08	39	E9	04	ED	07	39	28	AA	20	96	34
3A20	20	9B	35	20	7D	34	20	94	35	AD	03	30	38	C9	18	90
3A30	03	20	7D	34	20	0D	33	CE	06	39	10	EA	20	1A	33	20
3A40	72	34	60													

----------> FINE APPENDICE D6 <----------

D.7 Un disassemblatore (subroutine per il modo di indirizzamento)

```
APPENDICE D7:                 DISASSEMBLER
                    (SUBROUTINE DI INDIRIZZAMENTO)


 VEDI CAPITOLO 9


DUMP $3A50-$3B43

         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3A50    20 CF 39 A2 02 A9 04 60 20 50 3A 20 EB 39 A2 02
3A60    A9 06 60 20 50 3A 20 F6 39 A2 02 A9 06 60 A9 41
3A70    20 40 34 A2 00 A9 01 60 A2 00 A9 00 60 A9 23 20
3A80    40 34 A9 24 20 40 34 20 C8 39 A2 01 A9 04 60 20
3A90    E1 39 20 50 3A 20 E5 39 A9 06 A2 02 60 20 E1 39
3AA0    20 F3 3A 20 E5 39 A2 01 A9 06 60 20 E1 39 20 EB
3AB0    3A 20 E5 39 20 F6 39 A2 01 A9 06 60 20 0D 33 20
3AC0    12 35 20 95 32 48 20 0D 33 68 C9 00 10 03 CE 06
3AD0    32 08 D8 18 6D 05 32 90 03 EE 06 32 8D 05 32 28
3AE0    20 9B 35 20 2B 35 A2 01 A9 04 60 20 C8 39 A2 01
3AF0    A9 02 60 20 EB 3A 20 EB 39 A2 01 A9 04 60 20 EB
3B00    3A 20 F6 39 A2 01 A9 04 60 68 68 68 68 20 7D 37
3B10    30 0D 20 95 32 C9 FF F0 06 20 40 34 18 90 EE 20
3B20    72 34 20 7D 37 60 78 3A 6E 3A 7D 3A EB 3A F3 3A
3B30    FE 3A 50 3A 58 3A 63 3A 78 3A BC 3A 9D 3A AB 3A
3B40    8F 3A 09 3B

---------->  FINE APPENDICE D7  <----------
```

D.8 Un disassemblatore (tabelle)

APPENDICE D8: DISASSEMBLER (TABELLE)

VEDI CAPITOLO 9

DUMP $3B50-$3DFF

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3B50	7F	42	41	44	41	44	43	41	4E	44	41	53	4C	42	43	43
3B60	42	43	53	42	45	51	42	49	54	42	4D	49	42	4E	45	42
3B70	50	4C	42	52	4B	42	56	43	42	56	53	43	4C	43	43	4C
3B80	44	43	4C	49	43	4C	56	43	4D	50	43	50	58	43	50	59
3B90	44	45	43	44	45	58	44	45	59	45	4F	52	49	4E	43	49
3BA0	4E	58	49	4E	59	4A	4D	50	4A	53	52	4C	44	41	4C	44
3BB0	58	4C	44	59	4C	53	52	4E	4F	50	4F	52	41	50	48	41
3BC0	50	48	50	50	4C	41	50	4C	50	52	4F	4C	52	4F	52	52
3BD0	54	49	52	54	53	53	42	43	53	45	43	53	45	44	53	45
3BE0	49	53	54	41	53	54	58	53	54	59	54	41	58	54	41	59
3BF0	54	53	58	54	58	41	54	58	53	54	59	41	54	45	58	FF
3C00	22	6A	01	01	01	6A	0A	01	70	6A	0A	01	01	6A	0A	01
3C10	1F	6A	01	01	01	6A	0A	01	2B	6A	01	01	01	6A	0A	01
3C20	58	07	01	01	16	07	79	01	76	07	79	01	16	07	79	01
3C30	19	07	01	01	01	07	79	01	88	07	01	01	01	07	79	01
3C40	7F	49	01	01	01	49	64	01	6D	49	64	01	55	49	64	01
3C50	25	49	01	01	01	49	64	01	31	49	01	01	01	49	64	01
3C60	82	04	01	01	01	04	7C	01	73	04	7C	01	55	04	7C	01
3C70	28	04	01	01	01	04	7C	01	8E	04	01	01	01	04	7C	AC
3C80	01	91	01	01	97	91	94	01	46	01	A3	01	97	91	94	01
3C90	0D	91	01	01	97	91	94	01	A9	91	A3	01	01	91	01	01
3CA0	61	5B	5E	01	61	5B	5E	01	9D	5B	9A	01	61	5B	5E	01
3CB0	10	5B	01	01	61	5B	5E	01	34	5B	9E	01	61	5B	5E	01
3CC0	3D	37	01	01	3D	37	40	01	52	37	43	01	3D	37	40	01
3CD0	1C	37	01	01	01	37	40	01	2E	37	01	01	01	37	40	01
3CE0	3A	85	01	01	3A	85	4C	01	4F	85	67	01	3A	85	4C	01
3CF0	13	85	01	01	01	85	4C	01	8B	85	01	01	01	85	4C	01
3D00	12	16	00	00	00	06	06	00	12	04	02	00	00	0C	0C	00
3D10	14	18	00	00	00	0E	0E	00	12	10	00	00	00	16	16	00
3D20	0C	16	00	00	06	06	06	00	12	04	02	00	0C	0C	0C	00
3D30	14	18	00	00	00	08	08	00	12	10	00	00	00	0E	0E	00
3D40	12	16	00	00	00	06	06	00	12	0C	02	00	0C	0C	0C	00
3D50	14	18	00	00	00	08	08	00	12	10	00	00	00	0E	0E	00
3D60	12	16	00	00	00	06	06	00	12	04	02	00	1A	0C	0C	00
3D70	14	18	00	00	00	08	08	00	12	10	00	00	00	0E	0E	1C
3D80	00	16	00	00	06	06	06	00	12	00	12	00	0C	0C	0C	00
3D90	14	18	00	00	08	08	0A	00	12	10	12	00	00	0E	00	00
3DA0	04	16	04	00	06	06	06	00	12	04	12	00	0C	0C	0C	00
3DB0	14	18	00	00	08	08	0A	00	14	10	12	00	0E	0E	10	00
3DC0	04	16	00	00	06	06	06	00	12	04	12	00	0C	0C	0C	00
3DD0	14	18	00	00	00	08	08	00	12	10	00	00	00	0E	0E	00
3DE0	04	16	00	00	06	06	06	00	12	04	12	00	0C	0C	0C	00
3DF0	14	18	00	00	00	08	08	00	12	10	00	00	00	0E	0E	00

----------> FINE APPENDICE D8 <----------

D.9 Utility di trasferimento

APPENDICE D9: UTILITY DI TRASFERIMENTO

VEDI CAPITOLO 10

DUMP $37B0-$38EE

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
37B0	00	00	00	00	20	08	34	20	E4	34	7F	0D	0A	20	20	20
37C0	20	20	4D	4F	56	45	20	54	4F	4F	4C	2E	0D	0A	0A	FF
37D0	20	E3	35	20	B9	38	AE	55	35	38	AD	54	35	ED	52	35
37E0	8D	B0	37	B0	02	CA	38	8A	ED	53	35	8D	B1	37	B0	03
37F0	A9	00	60	A0	03	B9	FB	00	48	88	10	F9	38	AD	53	35
3800	CD	B3	37	90	40	D0	18	AD	52	35	CD	B2	37	90	36	D0
3810	0E	A0	00	68	99	FB	00	C8	C0	04	D0	F7	A9	FF	60	20
3820	A4	38	A0	00	AE	B1	37	F0	0E	B1	FB	91	FD	C8	D0	F9
3830	E6	FC	E6	FE	CA	D0	F2	88	C8	B1	FB	91	FD	CC	B0	37
3840	D0	F6	4C	11	38	AD	B1	37	F0	48	AC	B1	37	AD	B0	37
3850	38	E9	FF	B0	01	88	AA	84	FE	8A	18	6D	52	35	85	FB
3860	90	01	C8	98	6D	53	35	85	FC	8A	18	6D	B2	37	85	FD
3870	90	02	E6	FE	A5	FE	6D	B3	37	85	FE	AE	B1	37	A0	FF
3880	B1	FB	91	FD	88	D0	F9	B1	FB	91	FD	C6	FC	C6	FE	CA
3890	D0	EC	20	A4	38	AC	B0	37	B1	FB	91	FD	88	C0	FF	D0
38A0	F7	4C	11	38	AD	52	35	85	FB	AD	53	35	85	FC	AD	B2
38B0	37	85	FD	AD	B3	37	85	FE	60	20	08	34	20	E4	34	7F
38C0	0D	0A	53	45	54	20	44	45	53	54	49	4E	41	54	49	4F
38D0	4E	20	41	4E	44	20	50	52	45	53	53	20	51	2E	FF	20
38E0	07	32	AD	05	32	8D	B2	37	AD	06	32	8D	B3	37	60	

----------> FINE APPENDICE D9 <----------

D.10 Un semplice text editor

```
APPENDICE D10:            TEXT EDITOR


 VEDI CAPITOLO 11


DUMP $3E00-$3FFF

         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3E00    00 00 20 0F 3E 20 37 3E 20 C8 3E 18 18 90 F6 20
3E10    08 34 20 E4 34 7F 0D 0A 0A 53 45 54 20 55 50 20
3E20    45 44 49 54 20 42 55 46 46 45 52 2E 0D 0A 0A FF
3E30    20 E3 35 20 9A 37 60 20 C4 31 20 2B 31 AE 03 30
3E40    A0 03 20 13 31 20 2B 31 20 76 31 20 C4 31 20 5E
3E50    3E 20 D3 31 20 76 31 20 88 3E 20 D3 31 60 20 12
3E60    35 AD 03 30 4A AA CA 20 1A 33 CA 10 FA AD 03 30
3E70    8D 00 3E 20 95 32 20 9B 31 20 7F 31 20 0D 33 CE
3E80    00 3E 10 EF 20 2B 35 60 AD 03 30 4A E9 02 20 81
3E90    31 AD 01 3E C9 01 D0 05 A9 49 18 90 02 A9 4F 20
3EA0    9B 31 A9 02 20 81 31 AD 07 30 20 9B 31 A9 02 20
3EB0    81 31 AD 06 32 20 A3 31 AD 05 32 20 A3 31 60 00
3EC0    93 94 1D 9D 10 14 51 00 20 E0 32 CD C6 3E D0 17
3ED0    48 20 E0 32 CD C6 3E D0 04 68 68 68 60 8D C7 3E
3EE0    68 20 E7 3E AD C7 3E CD C1 3E D0 0B CE 01 3E 10
3EF0    05 A9 01 8D 01 3E 60 CD C2 3E D0 04 20 79 3F 60
3F00    CD C3 3E D0 04 20 87 3F 60 CD C5 3E D0 04 20 DD
3F10    3F 60 CD C4 3E D0 04 20 C5 3F 60 CD C0 3E D0 04
3F20    20 B4 3F 60 AE 01 3E F0 04 20 34 3F 60 20 2D 33
3F30    20 7D 37 60 48 20 12 35 AD 53 35 48 AD 52 35 48
3F40    AD 55 35 48 AD 54 35 48 20 61 36 20 7D 37 30 11
3F50    20 E2 38 AD 54 35 D0 03 CE 55 35 CE 54 35 20 D6
3F60    37 68 8D 54 35 68 8D 55 35 68 8D 52 35 68 8D 53
3F.70   35 20 2B 35 68 20 2D 3F 60 20 95 32 C9 FF F0 04
3F80    20 7D 37 60 A9 FF 60 38 AD 53 35 CD 06 32 90 0C
3F90    D0 10 AD 52 35 CD 05 32 F0 17 B0 06 20 1A 33 A9
3FA0    00 60 AD 52 35 8D 05 32 AD 53 35 8D 06 32 A9 00
3FB0    60 A9 FF 60 20 9A 37 A9 FF 20 2D 33 20 7D 37 10
3FC0    F6 20 9A 37 60 20 9A 37 20 14 34 20 95 32 C9 FF
3FD0    F0 08 20 40 34 20 7D 37 10 F1 4C 1A 34 20 12 35
3FE0    AD 53 35 48 AD 52 35 48 20 E2 38 20 7D 37 20 61
3FF0    36 20 D6 37 68 8D 52 35 68 8D 53 35 20 2B 35 60

---------->  FINE APPENDICE D10  <----------
```

D.11 Il Monitor Visibile integrato

```
APPENDICE D11:      MONITOR VISIBILE INTEGRATO


 VEDI CAPITOLO 12

DUMP $30B0-$30FC

         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

30B0    C9 50 D0 09 AD 00 34 49 FF 8D 00 34 60 C9 55 D0
30C0    09 AD 02 34 49 FF 8D 02 34 60 C9 48 D0 0D AD 00
30D0    34 D0 04 20 57 35 60 20 A8 35 60 C9 4D D0 04 20
30E0    B4 37 60 C9 3F D0 0D AD 00 34 D0 04 20 09 39 60
30F0    20 26 39 60 C9 54 D0 04 20 02 3E 60 60

----------> FINE APPENDICE D11  <----------
```

D.12 Dati di sistema per il VIC-20

```
APPENDICE D12:       DATI DI SISTEMA PER IL VIC-20


 VEDI APPENDICE B1

DUMP $3000-$30AF

         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3000    00 10 16 15 18 11 20 1E 35 30 3C 30 41 30 10 30
3010    60 48 A5 FC 48 18 69 84 85 FC A0 00 AD 86 02 91
3020    FB 68 85 FC 68 38 C9 40 90 0A C9 60 90 03 E9 20
3030    60 38 E9 40 60 20 E4 FF AA F0 FA 60 A2 01 4C 43
3040    30 A2 02 48 20 C9 FF 68 20 D2 FF 60 01 00 00 00
3050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3060    00 00 A9 03 AE 4C 30 A8 20 BA FF AD 4D 30 A2 4E
3070    A0 30 20 BD FF AD 52 35 85 FD AD 53 35 85 FE A9
3080    FD AE 54 35 AC 55 35 E8 D0 01 C8 20 D8 FF 60 A9
3090    00 48 28 20 BD FF A9 01 A2 00 A0 FF 20 BA FF 20
30A0    C0 FF A9 03 8D 51 35 20 07 32 A9 01 20 C3 FF 60

---------->  FINE APPENDICE D12  <----------
```

D.13 Dati di sistema per il C-64

```
APPENDICE D13:   DATI DI SISTEMA PER IL  COMMODORE 64


 VEDI APPENDICE B2

DUMP $3000-$30AA

         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F

3000    00 04 28 27 18 07 20 1E 35 30 3C 30 41 30 10 30
3010    60 48 A5 FC 48 18 69 D4 85 FC A0 00 AD 86 02 91
3020    FB 68 85 FC 68 38 C9 40 90 0A C9 60 90 03 E9 20
3030    60 38 E9 40 60 20 E4 FF AA F0 FA 60 A2 01 4C 43
3040    30 A2 02 48 20 C9 FF 68 20 D2 FF 60 01 00 00 00
3050    00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3060    00 00 A9 03 AE 4C 30 A8 20 BA FF AD 4D 30 A2 4E
3070    A0 30 20 BD FF AD 52 35 85 FD AD 53 35 85 FE A9
3080    FD AE 54 35 AC 55 35 E8 D0 01 C8 20 D8 FF 60 A9
3090    00 48 28 20 BD FF A9 01 A2 00 A0 FF 20 BA FF 20
30A0    C0 FF 20 07 32 A9 01 20 C3 FF 60

----------->  FINE APPENDICE D13  <----------
```

Appendice E

Dump decimali

E.1 Utility per la gestione dello schermo

```
APPENDICE E1:          UTILITY DELLO SCHERMO


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 12544 A 12769
DA USARE CON IL PROGRAMMA BASIC
LOADER.


1000  DATA  12544, 32, 196, 49, 32, 43, 49, 174, 3, 13122
1001  DATA  12552, 48, 172, 4, 48, 32, 19, 49, 32, 12956
1002  DATA  12560, 211, 49, 96, 142, 42, 49, 152, 170, 13471
1003  DATA  12568, 173, 6, 48, 172, 42, 49, 145, 251, 13454
1004  DATA  12576, 136, 16, 251, 32, 118, 49, 202, 16, 13396
1005  DATA  12584, 239, 96, 39, 162, 0, 160, 0, 24, 13304
1006  DATA  12592, 144, 10, 173, 4, 48, 74, 168, 173, 13386
1007  DATA  12600, 3, 48, 74, 170, 56, 236, 3, 48, 13238
1008  DATA  12608, 144, 3, 174, 3, 48, 56, 204, 4, 13244
1009  DATA  12616, 48, 144, 3, 172, 4, 48, 173, 0, 13208
1010  DATA  12624, 48, 133, 251, 173, 1, 48, 133, 252, 13663
1011  DATA  12632, 8, 216, 138, 24, 101, 251, 144, 3, 13517
1012  DATA  12640, 230, 252, 24, 192, 0, 240, 11, 24, 13613
1013  DATA  12648, 109, 2, 48, 144, 2, 230, 252, 136, 13571
1014  DATA  12656, 208, 245, 133, 251, 40, 96, 173, 2, 13804
1015  DATA  12664, 48, 24, 144, 5, 32, 155, 49, 169, 13290
1016  DATA  12672, 1, 8, 216, 24, 101, 251, 144, 2, 13419
1017  DATA  12680, 230, 252, 133, 251, 56, 173, 5, 48, 13828
1018  DATA  12688, 197, 252, 176, 5, 173, 1, 48, 133, 13673
1019  DATA  12696, 252, 40, 96, 32, 17, 48, 160, 0, 13341
1020  DATA  12704, 145, 251, 96, 72, 74, 74, 74, 74, 13564
```

```
1021  DATA  12712, 32, 182, 49, 32, 124, 49, 104, 32, 13316
1022  DATA  12720, 182, 49, 32, 124, 49, 96, 8, 216, 13476
1023  DATA  12728, 41, 15, 201, 10, 48, 2, 105, 6, 13156
1024  DATA  12736, 105, 48, 40, 96, 104, 170, 104, 168, 13571
1025  DATA  12744, 165, 252, 72, 165, 251, 72, 152, 72, 13945
1026  DATA  12752, 138, 72, 96, 104, 170, 104, 168, 104, 13708
1027  DATA  12760, 133, 251, 104, 133, 252, 152, 72, 138, 13995
1028  DATA  12768, 72, 96, 0, 255, 0, 255, 0, 255, 13701

---------->  FINE APPENDICE E1  <----------
```

E.2 Il Monitor Visibile
(primo livello e subroutine di visualizzazione)

```
APPENDICE E2:           MONITOR VISIBILE
              (PRIMO LIVELLO E SUBROUTINE VIDEO)


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 12800 A 13011
DA USARE CON IL PROGRAMMA BASIC
LOADER.


1100  DATA  12800, 0, 0, 0, 0, 0, 0, 0, 8, 12808
1101  DATA  12808, 216, 32, 18, 50, 32, 227, 50, 24, 13457
1102  DATA  12816, 144, 246, 32, 196, 49, 32, 37, 50, 13602
1103  DATA  12824, 32, 53, 50, 32, 93, 50, 32, 176, 13342
1104  DATA  12832, 50, 32, 211, 49, 96, 162, 0, 160, 13592
1105  DATA  12840, 0, 32, 60, 49, 174, 3, 48, 160, 13366
1106  DATA  12848, 3, 32, 19, 49, 96, 162, 11, 160, 13380
1107  DATA  12856, 0, 32, 60, 49, 160, 0, 140, 82, 13379
1108  DATA  12864, 50, 185, 83, 50, 32, 124, 49, 238, 13675
1109  DATA  12872, 82, 50, 172, 82, 50, 192, 10, 208, 13718
1110  DATA  12880, 240, 96, 10, 65, 32, 32, 88, 32, 13475
1111  DATA  12888, 32, 89, 32, 32, 80, 162, 0, 160, 13475
1112  DATA  12896, 1, 32, 60, 49, 173, 6, 50, 32, 13299
1113  DATA  12904, 163, 49, 173, 5, 50, 32, 163, 49, 13588
1114  DATA  12912, 32, 127, 49, 32, 149, 50, 72, 32, 13455
1115  DATA  12920, 163, 49, 32, 127, 49, 104, 32, 124, 13600
1116  DATA  12928, 49, 32, 127, 49, 162, 0, 189, 1, 13537
1117  DATA  12936, 50, 32, 163, 49, 32, 127, 49, 232, 13670
1118  DATA  12944, 224, 4, 208, 242, 96, 165, 251, 72, 14206
1119  DATA  12952, 166, 252, 173, 5, 50, 133, 251, 173, 14155
1120  DATA  12960, 6, 50, 133, 252, 160, 0, 177, 251, 13989
1121  DATA  12968, 168, 104, 133, 251, 134, 252, 152, 96, 14258
1122  DATA  12976, 172, 0, 50, 56, 192, 7, 144, 5, 13602
1123  DATA  12984, 160, 0, 140, 0, 50, 185, 205, 50, 13774
1124  DATA  12992, 170, 160, 2, 32, 60, 49, 173, 7, 13645
1125  DATA  13000, 48, 32, 124, 49, 96, 3, 6, 8, 13366
1126  DATA  13008, 11, 14, 17, 20, 0, 255, 0, 255, 13580

----------> FINE APPENDICE E2 <----------
```

E.3 Il Monitor Visibile
(subroutine di aggiornamento)

```
APPENDICE E3:          MONITOR VISIBILE
                      (SUBROUTINE AGGIOR)


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 13024 A 13294
DA USARE CON IL PROGRAMMA BASIC
LOADER.


1200  DATA  13024, 108, 8, 48, 32, 224, 50, 201, 29, 13724
1201  DATA  13032, 208, 16, 238, 0, 50, 173, 0, 50, 13767
1202  DATA  13040, 201, 7, 208, 5, 169, 0, 141, 0, 13771
1203  DATA  13048, 50, 96, 201, 157, 208, 11, 206, 0, 13977
1204  DATA  13056, 50, 16, 5, 169, 6, 141, 0, 50, 13493
1205  DATA  13064, 96, 201, 32, 208, 9, 238, 5, 50, 13903
1206  DATA  13072, 208, 3, 238, 6, 50, 96, 201, 13, 13887
1207  DATA  13080, 208, 12, 173, 5, 50, 208, 3, 206, 13945
1208  DATA  13088, 6, 50, 206, 5, 50, 96, 174, 0, 13675
1209  DATA  13096, 50, 224, 2, 208, 27, 168, 165, 251, 14191
1210  DATA  13104, 72, 166, 252, 173, 5, 50, 133, 251, 14206
1211  DATA  13112, 173, 6, 50, 133, 252, 152, 160, 0, 14038
1212  DATA  13120, 145, 251, 134, 252, 104, 133, 251, 96, 14486
1213  DATA  13128, 201, 71, 208, 35, 172, 3, 50, 174, 14042
1214  DATA  13136, 2, 50, 173, 4, 50, 72, 173, 1, 13661
1215  DATA  13144, 50, 40, 32, 108, 51, 8, 141, 1, 13575
1216  DATA  13152, 50, 142, 2, 50, 140, 3, 50, 104, 13693
1217  DATA  13160, 141, 4, 50, 96, 108, 5, 50, 72, 13686
1218  DATA  13168, 32, 213, 51, 48, 75, 168, 104, 152, 14011
1219  DATA  13176, 174, 0, 50, 208, 20, 162, 3, 24, 13817
1220  DATA  13184, 14, 5, 50, 46, 6, 50, 202, 16, 13573
1221  DATA  13192, 246, 152, 13, 5, 50, 141, 5, 50, 13854
1222  DATA  13200, 96, 224, 1, 208, 24, 41, 15, 72, 13881
1223  DATA  13208, 32, 149, 50, 10, 10, 10, 10, 41, 13520
1224  DATA  13216, 240, 141, 172, 51, 104, 13, 172, 51, 14160
1225  DATA  13224, 32, 45, 51, 96, 0, 202, 202, 202, 14054
1226  DATA  13232, 160, 3, 24, 30, 1, 50, 136, 16, 13652
1227  DATA  13240, 249, 29, 1, 50, 157, 1, 50, 96, 13873
1228  DATA  13248, 104, 201, 147, 208, 4, 32, 0, 49, 13993
1229  DATA  13256, 96, 201, 81, 208, 4, 104, 104, 40, 14094
1230  DATA  13264, 96, 32, 16, 48, 96, 56, 233, 48, 13889
1231  DATA  13272, 144, 15, 201, 10, 144, 14, 233, 7, 14040
1232  DATA  13280, 201, 16, 176, 5, 56, 201, 10, 176, 14121
1233  DATA  13288, 3, 169, 255, 96, 162, 0, 96, 255, 14324

---------->  FINE APPENDICE E3  <----------
```

E.4 Utility di stampa

```
APPENDICE E4:          UTILITY DI STAMPA



LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 13312 A 13635
DA USARE CON IL PROGRAMMA BASIC
LOADER.


1300  DATA  13312, 0, 255, 0, 0, 0, 0, 0, 0, 13567
1301  DATA  13320, 169, 255, 141, 1, 52, 96, 169, 0, 14203
1302  DATA  13328, 141, 1, 52, 96, 169, 255, 141, 0, 14183
1303  DATA  13336, 52, 96, 169, 0, 141, 0, 52, 96, 13942
1304  DATA  13344, 169, 255, 141, 2, 52, 96, 169, 0, 14228
1305  DATA  13352, 141, 2, 52, 96, 32, 8, 52, 32, 13767
1306  DATA  13360, 20, 52, 32, 32, 52, 96, 32, 14, 13690
1307  DATA  13368, 52, 32, 26, 52, 32, 38, 52, 96, 13748
1308  DATA  13376, 201, 0, 240, 36, 141, 3, 52, 173, 14222
1309  DATA  13384, 1, 52, 240, 6, 173, 3, 52, 32, 13943
1310  DATA  13392, 105, 52, 173, 0, 52, 240, 6, 173, 14193
1311  DATA  13400, 3, 52, 32, 108, 52, 173, 2, 52, 13874
1312  DATA  13408, 240, 6, 173, 3, 52, 32, 111, 52, 14077
1313  DATA  13416, 96, 108, 10, 48, 108, 12, 48, 108, 13954
1314  DATA  13424, 14, 48, 169, 13, 32, 64, 52, 169, 13985
1315  DATA  13432, 10, 32, 64, 52, 96, 169, 32, 32, 13919
1316  DATA  13440, 64, 52, 96, 72, 74, 74, 74, 74, 14020
1317  DATA  13448, 32, 182, 49, 32, 64, 52, 104, 32, 13995
1318  DATA  13456, 182, 49, 32, 64, 52, 96, 169, 32, 14132
1319  DATA  13464, 142, 4, 52, 72, 174, 4, 52, 240, 14204
1320  DATA  13472, 10, 206, 4, 52, 32, 64, 52, 104, 13996
1321  DATA  13480, 24, 144, 240, 104, 96, 142, 4, 52, 14286
1322  DATA  13488, 174, 4, 52, 240, 9, 206, 4, 52, 14229
1323  DATA  13496, 32, 114, 52, 24, 144, 242, 96, 142, 14342
1324  DATA  13504, 5, 52, 181, 1, 72, 181, 0, 72, 14068
1325  DATA  13512, 174, 5, 52, 161, 0, 201, 255, 240, 14600
1326  DATA  13520, 12, 246, 0, 208, 2, 246, 1, 32, 14267
1327  DATA  13528, 64, 52, 24, 144, 235, 104, 149, 0, 14300
1328  DATA  13536, 104, 149, 1, 96, 104, 170, 104, 168, 14432
1329  DATA  13544, 32, 18, 53, 142, 5, 50, 140, 6, 13990
1330  DATA  13552, 50, 32, 13, 51, 32, 13, 51, 32, 13826
1331  DATA  13560, 149, 50, 201, 255, 240, 6, 32, 64, 14557
1332  DATA  13568, 52, 24, 144, 240, 174, 5, 50, 172, 14429
1333  DATA  13576, 6, 50, 32, 43, 53, 152, 72, 138, 14122
1334  DATA  13584, 72, 96, 104, 141, 6, 52, 104, 141, 14300
1335  DATA  13592, 7, 52, 173, 6, 50, 72, 173, 5, 14130
1336  DATA  13600, 50, 72, 173, 7, 52, 72, 173, 6, 14205
1337  DATA  13608, 52, 72, 96, 104, 141, 6, 52, 104, 14235
1338  DATA  13616, 141, 7, 52, 104, 141, 5, 50, 104, 14220
1339  DATA  13624, 141, 6, 50, 173, 7, 52, 72, 173, 14298
1340  DATA  13632, 6, 52, 72, 96, 0, 255, 0, 255, 14368

---------->  FINE APPENDICE E4  <----------
```

E.5 Il dump esadecimale

```
APPENDICE E5:          DUMP ESADECIMALE



LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 13648 A 14246
DA USARE CON IL PROGRAMMA BASIC
LOADER.

1400  DATA  13648, 0, 7, 0, 0, 255, 255, 0, 32, 14197
1401  DATA  13656, 8, 52, 169, 4, 141, 80, 53, 173, 14336
1402  DATA  13664, 5, 50, 41, 240, 141, 5, 50, 32, 14228
1403  DATA  13672, 155, 53, 32, 125, 52, 32, 125, 52, 14298
1404  DATA  13680, 32, 148, 53, 32, 13, 51, 173, 5, 14187
1405  DATA  13688, 50, 45, 81, 53, 208, 239, 32, 114, 14510
1406  DATA  13696, 52, 173, 5, 50, 41, 15, 208, 3, 14243
1407  DATA  13704, 32, 114, 52, 206, 80, 53, 208, 215, 14664
1408  DATA  13712, 32, 14, 52, 96, 32, 149, 50, 32, 14169
1409  DATA  13720, 131, 52, 96, 173, 6, 50, 32, 131, 14391
1410  DATA  13728, 52, 173, 5, 50, 32, 131, 52, 96, 14319
1411  DATA  13736, 32, 195, 53, 32, 227, 53, 32, 154, 14514
1412  DATA  13744, 55, 32, 20, 52, 32, 229, 54, 32, 14250
1413  DATA  13752, 60, 55, 16, 251, 32, 114, 52, 32, 14364
1414  DATA  13760, 26, 52, 96, 32, 8, 52, 32, 26, 14084
1415  DATA  13768, 52, 32, 228, 52, 127, 13, 80, 82, 14434
1416  DATA  13776, 73, 78, 84, 73, 78, 71, 32, 72, 14337
1417  DATA  13784, 69, 88, 68, 85, 77, 80, 13, 10, 14274
1418  DATA  13792, 10, 255, 96, 32, 8, 52, 32, 228, 14505
1419  DATA  13800, 52, 127, 13, 10, 83, 69, 84, 32, 14270
1420  DATA  13808, 83, 84, 65, 82, 84, 73, 78, 71, 14428
1421  DATA  13816, 32, 65, 68, 68, 82, 69, 83, 83, 14366
1422  DATA  13824, 32, 65, 78, 68, 32, 80, 82, 69, 14330
1423  DATA  13832, 83, 83, 32, 34, 81, 34, 46, 255, 14480
1424  DATA  13840, 32, 7, 50, 32, 97, 54, 32, 8, 14152
1425  DATA  13848, 52, 32, 228, 52, 127, 13, 10, 83, 14445
1426  DATA  13856, 69, 84, 32, 69, 78, 68, 32, 65, 14353
1427  DATA  13864, 68, 68, 82, 69, 83, 83, 32, 65, 14414
1428  DATA  13872, 78, 68, 32, 80, 82, 69, 83, 83, 14447
1429  DATA  13880, 32, 34, 81, 34, 46, 255, 32, 7, 14401
1430  DATA  13888, 50, 56, 173, 6, 50, 205, 83, 53, 14564
1431  DATA  13896, 144, 36, 208, 8, 173, 5, 50, 205, 14725
1432  DATA  13904, 82, 53, 144, 26, 173, 6, 50, 141, 14579
1433  DATA  13912, 85, 53, 173, 5, 50, 141, 84, 53, 14556
1434  DATA  13920, 96, 173, 6, 50, 141, 83, 53, 173, 14695
1435  DATA  13928, 5, 50, 141, 82, 53, 96, 32, 228, 14615
1436  DATA  13936, 52, 127, 13, 10, 10, 10, 32, 69, 14259
1437  DATA  13944, 82, 82, 79, 82, 33, 33, 33, 32, 14400
1438  DATA  13952, 69, 78, 68, 32, 65, 68, 68, 82, 14482
1439  DATA  13960, 69, 83, 83, 32, 76, 69, 83, 83, 14538
1440  DATA  13968, 32, 84, 72, 65, 78, 32, 83, 84, 14498
1441  DATA  13976, 65, 82, 84, 32, 65, 68, 68, 82, 14522
1442  DATA  13984, 69, 83, 83, 44, 32, 87, 72, 73, 14527
1443  DATA  13992, 67, 72, 32, 73, 83, 32, 255, 32, 14638
1444  DATA  14000, 181, 54, 76, 22, 54, 169, 36, 32, 14624
1445  DATA  14008, 64, 52, 173, 83, 53, 32, 131, 52, 14648
1446  DATA  14016, 173, 82, 53, 32, 131, 52, 96, 169, 14804
1447  DATA  14024, 36, 32, 64, 52, 173, 85, 53, 32, 14551
1448  DATA  14032, 131, 52, 173, 84, 53, 32, 131, 52, 14740
```

```
1449  DATA  14040, 96, 32, 181, 54, 169, 45, 32, 64, 14713
1450  DATA  14048, 52, 32, 199, 54, 96, 32, 228, 52, 14793
1451  DATA  14056, 127, 13, 10, 10, 68, 85, 77, 80, 14526
1452  DATA  14064, 73, 78, 71, 32, 255, 32, 217, 54, 14876
1453  DATA  14072, 32, 114, 52, 32, 228, 52, 127, 10, 14719
1454  DATA  14080, 10, 32, 32, 32, 32, 32, 32, 32, 14314
1455  DATA  14088, 32, 48, 32, 32, 49, 32, 32, 50, 14395
1456  DATA  14096, 32, 32, 51, 32, 32, 52, 32, 32, 14391
1457  DATA  14104, 53, 32, 32, 54, 32, 32, 55, 32, 14426
1458  DATA  14112, 32, 56, 32, 32, 57, 32, 32, 65, 14450
1459  DATA  14120, 32, 32, 66, 32, 32, 67, 32, 32, 14445
1460  DATA  14128, 68, 32, 32, 69, 32, 32, 70, 13, 14476
1461  DATA  14136, 10, 10, 255, 96, 32, 114, 52, 173, 14878
1462  DATA  14144, 5, 50, 72, 41, 15, 141, 86, 53, 14607
1463  DATA  14152, 104, 41, 240, 141, 5, 50, 32, 155, 14920
1464  DATA  14160, 53, 162, 3, 32, 150, 52, 173, 86, 14871
1465  DATA  14168, 53, 240, 13, 162, 3, 32, 150, 52, 14873
1466  DATA  14176, 32, 13, 51, 206, 86, 53, 208, 243, 15068
1467  DATA  14184, 32, 148, 53, 32, 125, 52, 32, 125, 14783
1468  DATA  14192, 55, 48, 9, 173, 5, 50, 41, 15, 14588
1469  DATA  14200, 201, 0, 208, 236, 96, 56, 173, 6, 15176
1470  DATA  14208, 50, 205, 85, 53, 144, 11, 208, 15, 14979
1471  DATA  14216, 56, 173, 5, 50, 205, 84, 53, 176, 15018
1472  DATA  14224, 6, 32, 13, 51, 169, 0, 96, 169, 14760
1473  DATA  14232, 255, 96, 173, 82, 53, 141, 5, 50, 15087
1474  DATA  14240, 173, 83, 53, 141, 6, 50, 96, 255, 15097

---------->  FINE APPENDICE E5  <----------
```

E.6 Un disassemblatore (primo livello e subroutine di utility)

```
APPENDICE E6:          DISASSEMBLATORE (PRIMO LIVELLO E UTILITY)


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 14592 A 14914
DA USARE CON IL PROGRAMMA BASIC
LOADER.

1500  DATA  14592, 5, 0, 0, 0, 0, 0, 0, 0, 14597
1501  DATA  14600,11,32, 8, 52, 173, 0, 57, 141, 15074
1502  DATA  14608, 1, 57, 169, 255, 141, 84, 53, 141, 15509
1503  DATA  14616, 85, 53, 32, 114, 52, 32, 125, 57, 15166
1504  DATA  14624, 206, 1, 57, 208, 248, 96, 32, 26, 15498
1505  DATA  14632, 52, 32, 8, 52, 32, 228, 52, 127, 15215
1506  DATA  14640, 13, 10, 32, 32, 32, 32, 32, 80, 14903
1507  DATA  14648, 82, 73, 78, 84, 73, 78, 71, 32, 15219
1508  DATA  14656, 68, 73, 83, 65, 83, 83, 69, 77, 15257
1509  DATA  14664, 66, 76, 69, 82, 46, 13, 10, 255, 15281
1510  DATA  14672, 32, 227, 53, 32, 20, 52, 32, 228, 15348
1511  DATA  14680, 52, 127, 13, 10, 68, 73, 83, 65, 15161
1512  DATA  14688, 83, 83, 69, 77, 66, 76, 73, 78, 15293
1513  DATA  14696, 71, 32, 255, 32, 217, 54, 32, 154, 15543
1514  DATA  14704, 55, 32, 114, 52, 32, 125, 57, 16, 15187
1515  DATA  14712, 251, 32, 26, 52, 96, 32, 149, 50, 15400
1516  DATA  14720, 72, 32, 146, 57, 32, 125, 52, 104, 15340
1517  DATA  14728, 32, 175, 57, 32, 1, 58, 32, 125, 15240
1518  DATA  14736, 55, 96, 162, 3, 142, 2, 57, 170, 15423
1519  DATA  14744, 189, 0, 60, 170, 189, 80, 59, 142, 15633
1520  DATA  14752, 3, 57, 32, 64, 52, 174, 3, 57, 15194
1521  DATA  14760, 232, 206, 2, 57, 208, 238, 96, 170, 15969
1522  DATA  14768, 189, 0, 61, 170, 32, 184, 57, 96, 15557
1523  DATA  14776, 189, 38, 59, 141, 4, 57, 232, 189, 15685
1524  DATA  14784, 38, 59, 141, 5, 57, 108, 4, 57, 15253
1525  DATA  14792, 32, 13, 51, 32, 148, 53, 96, 32, 15249
1526  DATA  14800, 13, 51, 32, 149, 50, 72, 32, 13, 15212
1527  DATA  14808, 51, 32, 148, 53, 104, 32, 131, 52, 15411
1528  DATA  14816, 96, 169, 40, 208, 2, 169, 41, 32, 15573
1529  DATA  14824, 64, 52, 96, 169, 44, 32, 64, 52, 15397
1530  DATA  14832, 169, 88, 32, 64, 52, 96, 169, 44, 15546
1531  DATA  14840, 32, 64, 52, 169, 89, 32, 64, 52, 15394
1532  DATA  14848, 96, 141, 7, 57, 142, 6, 57, 202, 15556
1533  DATA  14856, 48, 6, 32, 26, 51, 202, 16, 250, 15487
1534  DATA  14864, 8, 216, 56, 173, 8, 57, 233, 4, 15619
1535  DATA  14872, 237, 7, 57, 40, 170, 32, 150, 52, 15617
1536  DATA  14880, 32, 155, 53, 32, 125, 52, 32, 148, 15509
1537  DATA  14888, 53, 173, 3, 48, 56, 201, 24, 144, 15590
1538  DATA  14896, 3, 32, 125, 52, 32, 13, 51, 206, 15410
1539  DATA  14904, 6, 57, 16, 234, 32, 26, 51, 32, 15358
1540  DATA  14912, 114, 52, 96, 91, 0, 255, 0, 255, 15775

----------> FINE APPENDICE E6 <-------------
```

E.7 Un disassemblatore (subroutine per il modo di indirizzamento)

```
APPENDICE E7:      DISASSEMBLATORE (SUBROUTINE DI INDIRIZZAMENTO)


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 14928 A 15171
DA USARE CON IL PROGRAMMA BASIC
LOADER.

1600  DATA  14928, 32, 207, 57, 162, 2, 169, 4, 96, 15657
1601  DATA  14936, 32, 80, 58, 32, 235, 57, 162, 2, 15594
1602  DATA  14944, 169, 6, 96, 32, 80, 58, 32, 246, 15663
1603  DATA  14952, 57, 162, 2, 169, 6, 96, 169, 65, 15678
1604  DATA  14960, 32, 64, 52, 162, 0, 169, 1, 96, 15536
1605  DATA  14968, 162, 0, 169, 0, 96, 169, 35, 32, 15631
1606  DATA  14976, 64, 52, 169, 36, 32, 64, 52, 32, 15477
1607  DATA  14984, 200, 57, 162, 1, 169, 4, 96, 32, 15705
1608  DATA  14992, 225, 57, 32, 80, 58, 32, 229, 57, 15762
1609  DATA  15000, 169, 6, 162, 2, 96, 32, 225, 57, 15749
1610  DATA  15008, 32, 243, 58, 32, 229, 57, 162, 1, 15822
1611  DATA  15016, 169, 6, 96, 32, 225, 57, 32, 235, 15868
1612  DATA  15024, 58, 32, 229, 57, 32, 246, 57, 162, 15897
1613  DATA  15032, 1, 169, 6, 96, 32, 13, 51, 32, 15432
1614  DATA  15040, 18, 53, 32, 149, 50, 72, 32, 13, 15459
1615  DATA  15048, 51, 104, 201, 0, 16, 3, 206, 6, 15635
1616  DATA  15056, 50, 8, 216, 24, 109, 5, 50, 144, 15662
1617  DATA  15064, 3, 238, 6, 50, 141, 5, 50, 40, 15597
1618  DATA  15072, 32, 155, 53, 32, 43, 53, 162, 1, 15603
1619  DATA  15080, 169, 4, 96, 32, 200, 57, 162, 1, 15801
1620  DATA  15088, 169, 2, 96, 32, 235, 58, 32, 235, 15947
1621  DATA  15096, 57, 162, 1, 169, 4, 96, 32, 235, 15852
1622  DATA  15104, 58, 32, 246, 57, 162, 1, 169, 4, 15833
1623  DATA  15112, 96, 104, 104, 104, 104, 32, 125, 55, 15836
1624  DATA  15120, 48, 13, 32, 149, 50, 201, 255, 240, 16108
1625  DATA  15128, 6, 32, 64, 52, 24, 144, 238, 32, 15720
1626  DATA  15136, 114, 52, 32, 125, 55, 96, 120, 58, 15788
1627  DATA  15144, 110, 58, 125, 58, 235, 58, 243, 58, 16089
1628  DATA  15152, 254, 58, 80, 58, 88, 58, 99, 58, 15905
1629  DATA  15160, 120, 58, 188, 58, 157, 58, 171, 58, 16028
1630  DATA  15168, 143, 58, 9, 59, 0, 255, 0, 255, 15947

---------->  FINE APPENDICE E7  <----------
```

E.8 Un disassemblatore (tabelle)

```
APPENDICE E8:          DISASSEMBLATORE (TABELLE)


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 15184 A 15871
DA USARE CON IL PROGRAMMA BASIC
LOADER.

1700  DATA  15184, 127, 66, 65, 68, 65, 68, 67, 65, 15775
1701  DATA  15192, 78, 68, 65, 83, 76, 66, 67, 67, 15762
1702  DATA  15200, 66, 67, 83, 66, 69, 81, 66, 73, 15771
1703  DATA  15208, 84, 66, 77, 73, 66, 78, 69, 66, 15787
1704  DATA  15216, 80, 76, 66, 82, 75, 66, 86, 67, 15814
1705  DATA  15224, 66, 86, 83, 67, 76, 67, 67, 76, 15812
1706  DATA  15232, 68, 67, 76, 73, 67, 76, 86, 67, 15812
1707  DATA  15240, 77, 80, 67, 80, 88, 67, 80, 89, 15868
1708  DATA  15248, 68, 69, 67, 68, 69, 88, 68, 69, 15814
1709  DATA  15256, 89, 69, 79, 82, 73, 78, 67, 73, 15866
1710  DATA  15264, 78, 88, 73, 78, 89, 74, 77, 80, 15901
1711  DATA  15272, 74, 83, 82, 76, 68, 65, 76, 68, 15864
1712  DATA  15280, 88, 76, 68, 89, 76, 83, 82, 78, 15920
1713  DATA  15288, 79, 80, 79, 82, 65, 80, 72, 65, 15890
1714  DATA  15296, 80, 72, 80, 80, 76, 65, 80, 76, 15905
1715  DATA  15304, 80, 82, 79, 76, 82, 79, 82, 82, 15946
1716  DATA  15312, 84, 73, 82, 84, 83, 83, 66, 67, 15934
1717  DATA  15320, 83, 69, 67, 83, 69, 68, 83, 69, 15911
1718  DATA  15328, 73, 83, 84, 65, 83, 84, 88, 83, 15971
1719  DATA  15336, 84, 89, 84, 65, 88, 84, 65, 89, 15984
1720  DATA  15344, 84, 83, 88, 84, 88, 65, 84, 88, 16008
1721  DATA  15352, 83, 84, 89, 65, 84, 69, 88, 255, 16169
1722  DATA  15360, 34, 106, 1, 1, 1, 106, 10, 1, 15620
1723  DATA  15368, 112, 106, 10, 1, 1, 106, 10, 1, 15715
1724  DATA  15376, 31, 106, 1, 1, 1, 106, 10, 1, 15633
1725  DATA  15384, 43, 106, 1, 1, 1, 106, 10, 1, 15653
1726  DATA  15392, 88, 7, 1, 1, 22, 7, 121, 1, 15640
1727  DATA  15400, 118, 7, 121, 1, 22, 7, 121, 1, 15798
1728  DATA  15408, 25, 7, 1, 1, 1, 7, 121, 1, 15572
1729  DATA  15416, 136, 7, 1, 1, 1, 7, 121, 1, 15691
1730  DATA  15424, 127, 73, 1, 1, 1, 73, 100, 1, 15801
1731  DATA  15432, 109, 73, 100, 1, 85, 73, 100, 1, 15974
1732  DATA  15440, 37, 73, 1, 1, 1, 73, 100, 1, 15727
1733  DATA  15448, 49, 73, 1, 1, 1, 73, 100, 1, 15747
1734  DATA  15456, 130, 4, 1, 1, 1, 4, 124, 1, 15722
1735  DATA  15464, 115, 4, 124, 1, 85, 4, 124, 1, 15922
1736  DATA  15472, 40, 4, 1, 1, 1, 4, 124, 1, 15648
1737  DATA  15480, 142, 4, 1, 1, 1, 4, 124, 172, 15929
1738  DATA  15488, 1, 145, 1, 1, 151, 145, 148, 1, 16081
1739  DATA  15496, 70, 1, 163, 1, 151, 145, 148, 1, 16176
1740  DATA  15504, 13, 145, 1, 1, 151, 145, 148, 1, 16109
1741  DATA  15512, 169, 145, 163, 1, 1, 145, 1, 1, 16138
1742  DATA  15520, 97, 91, 94, 1, 97, 91, 94, 1, 16086
1743  DATA  15528, 157, 91, 154, 1, 97, 91, 94, 1, 16214
1744  DATA  15536, 16, 91, 1, 1, 97, 91, 94, 1, 15928
1745  DATA  15544, 52, 91, 158, 1, 97, 91, 94, 1, 16129
1746  DATA  15552, 61, 55, 1, 1, 61, 55, 64, 1, 15851
```

```
1747  DATA  15560, 82, 55, 67, 1, 61, 55, 64, 1, 15946
1748  DATA  15568, 28, 55, 1, 1, 1, 55, 64, 1, 15774
1749  DATA  15576, 46, 55, 1, 1, 1, 55, 64, 1, 15800
1750  DATA  15584, 58, 133, 1, 1, 58, 133, 76, 1, 16045
1751  DATA  15592, 79, 133, 103, 1, 58, 133, 76, 1, 16176
1752  DATA  15600, 19, 133, 1, 1, 1, 133, 76, 1, 15965
1753  DATA  15608, 139, 133, 1, 1, 1, 133, 76, 1, 16093
1754  DATA  15616, 18, 22, 0, 0, 0, 6, 6, 0, 15668
1755  DATA  15624, 18, 4, 2, 0, 0, 12, 12, 0, 15672
1756  DATA  15632, 20, 24, 0, 0, 0, 14, 14, 0, 15704
1757  DATA  15640, 18, 16, 0, 0, 0, 22, 22, 0, 15718
1758  DATA  15648, 12, 22, 0, 0, 6, 6, 6, 0, 15700
1759  DATA  15656, 18, 4, 2, 0, 12, 12, 12, 0, 15716
1760  DATA  15664, 20, 24, 0, 0, 0, 8, 8, 0, 15724
1761  DATA  15672, 18, 16, 0, 0, 0, 14, 14, 0, 15734
1762  DATA  15680, 18, 22, 0, 0, 0, 6, 6, 0, 15732
1763  DATA  15688, 18, 12, 2, 0, 12, 12, 12, 0, 15756
1764  DATA  15696, 20, 24, 0, 0, 0, 8, 8, 0, 15756
1765  DATA  15704, 18, 16, 0, 0, 0, 14, 14, 0, 15766
1766  DATA  15712, 18, 22, 0, 0, 0, 6, 6, 0, 15764
1767  DATA  15720, 18, 4, 2, 0, 26, 12, 12, 0, 15794
1768  DATA  15728, 20, 24, 0, 0, 0, 8, 8, 0, 15788
1769  DATA  15736, 18, 16, 0, 0, 0, 14, 14, 28, 15826
1770  DATA  15744, 0, 22, 0, 0, 6, 6, 6, 0, 15784
1771  DATA  15752, 18, 0, 18, 0, 12, 12, 12, 0, 15824
1772  DATA  15760, 20, 24, 0, 0, 8, 8, 10, 0, 15830
1773  DATA  15768, 18, 16, 18, 0, 0, 14, 0, 0, 15834
1774  DATA  15776, 4, 22, 4, 0, 6, 6, 6, 0, 15824
1775  DATA  15784, 18, 4, 18, 0, 12, 12, 12, 0, 15860
1776  DATA  15792, 20, 24, 0, 0, 8, 8, 10, 0, 15862
1777  DATA  15800, 20, 16, 18, 0, 14, 14, 16, 0, 15898
1778  DATA  15808, 4, 22, 0, 0, 6, 6, 6, 0, 15852
1779  DATA  15816, 18, 4, 18, 0, 12, 12, 12, 0, 15892
1780  DATA  15824, 20, 24, 0, 0, 0, 8, 8, 0, 15884
1781  DATA  15832, 18, 16, 0, 0, 0, 14, 14, 0, 15894
1782  DATA  15840, 4, 22, 0, 0, 6, 6, 6, 0, 15884
1783  DATA  15848, 18, 4, 18, 0, 12, 12, 12, 0, 15924
1784  DATA  15856, 20, 24, 0, 0, 0, 8, 8, 0, 15916
1785  DATA  15864, 18, 16, 0, 0, 0, 14, 14, 0, 15926

---------->  FINE APPENDICE E8  <----------
```

E.9 Utility di trasferimento

```
APPENDICE E9:          UTILITY DI TRASFERIMENTO


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 14256 A 14574
DA USARE CON IL PROGRAMMA BASIC
LOADER.

1800  DATA  14256, 0, 0, 0, 0, 32, 8, 52, 32, 14380
1801  DATA  14264, 228, 52, 127, 13, 10, 32, 32, 32, 14790
1802  DATA  14272, 32, 32, 77, 79, 86, 69, 32, 84, 14763
1803  DATA  14280, 79, 79, 76, 46, 13, 10, 10, 255, 14848
1804  DATA  14288, 32, 227, 53, 32, 185, 56, 174, 85, 15132
1805  DATA  14296, 53, 56, 173, 84, 53, 237, 82, 53, 15087
1806  DATA  14304, 141, 176, 55, 176, 2, 202, 56, 138, 15250
1807  DATA  14312, 237, 83, 53, 141, 177, 55, 176, 3, 15237
1808  DATA  14320, 169, 0, 96, 160, 3, 185, 251, 0, 15184
1809  DATA  14328, 72, 136, 16, 249, 56, 173, 83, 53, 15166
1810  DATA  14336, 205, 179, 55, 144, 64, 208, 24, 173, 15388
1811  DATA  14344, 82, 53, 205, 178, 55, 144, 54, 208, 15323
1812  DATA  14352, 14, 160, 0, 104, 153, 251, 0, 200, 15234
1813  DATA  14360, 192, 4, 208, 247, 169, 255, 96, 32, 15563
1814  DATA  14368, 164, 56, 160, 0, 174, 177, 55, 240, 15394
1815  DATA  14376, 14, 177, 251, 145, 253, 200, 208, 249, 15873
1816  DATA  14384, 230, 252, 230, 254, 202, 208, 242, 136, 16138
1817  DATA  14392, 200, 177, 251, 145, 253, 204, 176, 55, 15853
1818  DATA  14400, 208, 246, 76, 17, 56, 173, 177, 55, 15408
1819  DATA  14408, 240, 72, 172, 177, 55, 173, 176, 55, 15528
1820  DATA  14416, 56, 233, 255, 176, 1, 136, 170, 132, 15575
1821  DATA  14424, 254, 138, 24, 109, 82, 53, 133, 251, 15468
1822  DATA  14432, 144, 1, 200, 152, 109, 83, 53, 133, 15307
1823  DATA  14440, 252, 138, 24, 109, 178, 55, 133, 253, 15582
1824  DATA  14448, 144, 2, 230, 254, 165, 254, 109, 179, 15785
1825  DATA  14456, 55, 133, 254, 174, 177, 55, 160, 255, 15719
1826  DATA  14464, 177, 251, 145, 253, 136, 208, 249, 177, 16060
1827  DATA  14472, 251, 145, 253, 198, 252, 198, 254, 202, 16225
1828  DATA  14480, 208, 236, 32, 164, 56, 172, 176, 55, 15579
1829  DATA  14488, 177, 251, 145, 253, 136, 192, 255, 208, 16105
1830  DATA  14496, 247, 76, 17, 56, 173, 82, 53, 133, 15333
1831  DATA  14504, 251, 173, 83, 53, 133, 252, 173, 178, 15800
1832  DATA  14512, 55, 133, 253, 173, 179, 55, 133, 254, 15747
1833  DATA  14520, 96, 32, 8, 52, 32, 228, 52, 127, 15147
1834  DATA  14528, 13, 10, 83, 69, 84, 32, 68, 69, 14956
1835  DATA  14536, 83, 84, 73, 78, 65, 84, 73, 79, 15155
1836  DATA  14544, 78, 32, 65, 78, 68, 32, 80, 82, 15059
1837  DATA  14552, 69, 83, 83, 32, 81, 46, 255, 32, 15233
1838  DATA  14560, 7, 50, 173, 5, 50, 141, 178, 55, 15219
1839  DATA  14568, 173, 6, 50, 141, 179, 55, 96, 255, 15523

---------->  FINE APPENDICE E9  <----------
```

E.10 Un semplice text editor

```
APPENDICE E10:          TEXT EDITOR



LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 15872 A 16383
DA USARE CON IL PROGRAMMA BASIC
LOADER.

1900  DATA  15872, 0, 0, 32, 15, 62, 32, 55, 62, 16130
1901  DATA  15880, 32, 200, 62, 24, 24, 144, 246, 32, 16644
1902  DATA  15888, 8, 52, 32, 228, 52, 127, 13, 10, 16410
1903  DATA  15896, 10, 83, 69, 84, 32, 85, 80, 32, 16371
1904  DATA  15904, 69, 68, 73, 84, 32, 66, 85, 70, 16451
1905  DATA  15912, 70, 69, 82, 46, 13, 10, 10, 255, 16467
1906  DATA  15920, 32, 227, 53, 32, 154, 55, 96, 32, 16601
1907  DATA  15928, 196, 49, 32, 43, 49, 174, 3, 48, 16522
1908  DATA  15936, 160, 3, 32, 19, 49, 32, 43, 49, 16323
1909  DATA  15944, 32, 118, 49, 32, 196, 49, 32, 94, 16546
1910  DATA  15952, 62, 32, 211, 49, 32, 118, 49, 32, 16537
1911  DATA  15960, 136, 62, 32, 211, 49, 96, 32, 18, 16596
1912  DATA  15968, 53, 173, 3, 48, 74, 170, 202, 32, 16723
1913  DATA  15976, 26, 51, 202, 16, 250, 173, 3, 48, 16745
1914  DATA  15984, 141, 0, 62, 32, 149, 50, 32, 155, 16605
1915  DATA  15992, 49, 32, 127, 49, 32, 13, 51, 206, 16551
1916  DATA  16000, 0, 62, 16, 239, 32, 43, 53, 96, 16541
1917  DATA  16008, 173, 3, 48, 74, 233, 2, 32, 129, 16702
1918  DATA  16016, 49, 173, 1, 62, 201, 1, 208, 5, 16716
1919  DATA  16024, 169, 73, 24, 144, 2, 169, 79, 32, 16716
1920  DATA  16032, 155, 49, 169, 2, 32, 129, 49, 173, 16790
1921  DATA  16040, 7, 48, 32, 155, 49, 169, 2, 32, 16534
1922  DATA  16048, 129, 49, 173, 6, 50, 32, 163, 49, 16699
1923  DATA  16056, 173, 5, 50, 32, 163, 49, 96, 255, 16879
1924  DATA  16064, 147, 148, 29, 157, 16, 20, 81, 0, 16662
1925  DATA  16072, 32, 224, 50, 205, 198, 62, 208, 23, 17074
1926  DATA  16080, 72, 32, 224, 50, 205, 198, 62, 208, 17131
1927  DATA  16088, 4, 104, 104, 104, 96, 141, 199, 62, 16902
1928  DATA  16096, 104, 32, 231, 62, 173, 199, 62, 205, 17164
1929  DATA  16104, 193, 62, 208, 11, 206, 1, 62, 16, 16863
1930  DATA  16112, 5, 169, 1, 141, 1, 62, 96, 205, 16792
1931  DATA  16120, 194, 62, 208, 4, 32, 121, 63, 96, 16900
1932  DATA  16128, 205, 195, 62, 208, 4, 32, 135, 63, 17032
1933  DATA  16136, 96, 205, 197, 62, 208, 4, 32, 221, 17161
1934  DATA  16144, 63, 96, 205, 196, 62, 208, 4, 32, 17010
1935  DATA  16152, 197, 63, 96, 205, 192, 62, 208, 4, 17179
1936  DATA  16160, 32, 180, 63, 96, 174, 1, 62, 240, 17008
1937  DATA  16168, 4, 32, 52, 63, 96, 32, 45, 51, 16543
1938  DATA  16176, 32, 125, 55, 96, 72, 32, 18, 53, 16659
1939  DATA  16184, 173, 83, 53, 72, 173, 82, 53, 72, 16945
1940  DATA  16192, 173, 85, 53, 72, 173, 84, 53, 72, 16957
1941  DATA  16200, 32, 97, 54, 32, 125, 55, 48, 17, 16660
1942  DATA  16208, 32, 226, 56, 173, 84, 53, 208, 3, 17043
1943  DATA  16216, 206, 85, 53, 206, 84, 53, 32, 214, 17149
1944  DATA  16224, 55, 104, 141, 84, 53, 104, 141, 85, 16991
1945  DATA  16232, 53, 104, 141, 82, 53, 104, 141, 83, 16993
1946  DATA  16240, 53, 32, 43, 53, 104, 32, 45, 63, 16665
1947  DATA  16248, 96, 32, 149, 50, 201, 255, 240, 4, 17275
1948  DATA  16256, 32, 125, 55, 96, 169, 255, 96, 56, 17140
```

```
1949  DATA  16264, 173, 83, 53, 205, 6, 50, 144, 12, 16990
1950  DATA  16272, 208, 16, 173, 82, 53, 205, 5, 50, 17064
1951  DATA  16280, 240, 23, 176, 6, 32, 26, 51, 169, 17003
1952  DATA  16288, 0, 96, 173, 82, 53, 141, 5, 50, 16888
1953  DATA  16296, 173, 83, 53, 141, 6, 50, 169, 0, 16971
1954  DATA  16304, 96, 169, 255, 96, 32, 154, 55, 169, 17330
1955  DATA  16312, 255, 32, 45, 51, 32, 125, 55, 16, 16923
1956  DATA  16320, 246, 32, 154, 55, 96, 32, 154, 55, 17144
1957  DATA  16328, 32, 20, 52, 32, 149, 50, 201, 255, 17119
1958  DATA  16336, 240, 8, 32, 64, 52, 32, 125, 55, 16944
1959  DATA  16344, 16, 241, 76, 26, 52, 32, 18, 53, 16858
1960  DATA  16352, 173, 83, 53, 72, 173, 82, 53, 72, 17113
1961  DATA  16360, 32, 226, 56, 32, 125, 55, 32, 97, 17015
1962  DATA  16368, 54, 32, 214, 55, 104, 141, 82, 53, 17103
1963  DATA  16376, 104, 141, 83, 53, 32, 43, 53, 96, 16981

---------->  FINE APPENDICE E10  <----------
```

E.11 Il Monitor Visibile integrato

```
APPENDICE E11:          MONITOR VISIBILE INTEGRATO



LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 12464 A 12540
DA USARE CON IL PROGRAMMA BASIC
LOADER.

2000  DATA  12464, 201, 80, 208, 9, 173, 0, 52, 73, 13260
2001  DATA  12472, 255, 141, 0, 52, 96, 201, 85, 208, 13510
2002  DATA  12480, 9, 173, 2, 52, 73, 255, 141, 2, 13187
2003  DATA  12488, 52, 96, 201, 72, 208, 13, 173, 0, 13303
2004  DATA  12496, 52, 208, 4, 32, 87, 53, 96, 32, 13060
2005  DATA  12504, 168, 53, 96, 201, 77, 208, 4, 32, 13343
2006  DATA  12512, 180, 55, 96, 201, 63, 208, 13, 173, 13501
2007  DATA  12520, 0, 52, 208, 4, 32, 9, 57, 96, 12978
2008  DATA  12528, 32, 38, 57, 96, 201, 84, 208, 4, 13248
2009  DATA  12536, 32, 2, 62, 96, 96, 126, 145, 154, 13249

---------->  FINE APPENDICE E11  <----------
```

E.12 Dati di sistema per il VIC-20

```
APPENDICE E12:          DATI DI SISTEMA PER IL VIC-20


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 12288 A 12458
DA USARE CON IL PROGRAMMA BASIC
LOADER.

2100  DATA  12288, 0, 16, 22, 21, 24, 17, 32, 30, 12450
2101  DATA  12296, 53, 48, 60, 48, 65, 48, 16, 48, 12682
2102  DATA  12304, 96, 72, 165, 252, 72, 24, 105, 132, 13222
2103  DATA  12312, 133, 252, 160, 0, 173, 134, 2, 145, 13311
2104  DATA  12320, 251, 104, 133, 252, 104, 56, 201, 64, 13485
2105  DATA  12328, 144, 10, 201, 96, 144, 3, 233, 32, 13191
2106  DATA  12336, 96, 56, 233, 64, 96, 32, 228, 255, 13396
2107  DATA  12344, 170, 240, 250, 96, 162, 1, 76, 67, 13406
2108  DATA  12352, 48, 162, 2, 72, 32, 201, 255, 104, 13228
2109  DATA  12360, 32, 210, 255, 96, 1, 0, 0, 0, 12954
2110  DATA  12368, 0, 0, 0, 0, 0, 0, 0, 0, 12368
2111  DATA  12376, 0, 0, 0, 0, 0, 0, 0, 0, 12376
2112  DATA  12384, 0, 0, 169, 3, 174, 76, 48, 168, 13022
2113  DATA  12392, 32, 186, 255, 173, 77, 48, 162, 78, 13403
2114  DATA  12400, 160, 48, 32, 189, 255, 173, 82, 53, 13392
2115  DATA  12408, 133, 253, 173, 83, 53, 133, 254, 169, 13659
2116  DATA  12416, 253, 174, 84, 53, 172, 85, 53, 232, 13522
2117  DATA  12424, 208, 1, 200, 32, 216, 255, 96, 169, 13601
2118  DATA  12432, 0, 72, 40, 32, 189, 255, 169, 1, 13190
2119  DATA  12440, 162, 0, 160, 255, 32, 186, 255, 32, 13522
2120  DATA  12448, 192, 255, 169, 3, 141, 81, 53, 32, 13374
2121  DATA  12456, 7, 50, 169, 1, 32, 195, 255, 96, 13261

---------->  FINE APPENDICE E12  <----------
```

E.13 Dati di sistema per il C-64

```
APPENDICE E13:        DATI DI SISTEMA PER IL COMMODORE 64


LE SEGUENTI ISTRUZIONI CONTENGONO IL
CODICE OGGETTO DECIMALE E LE CHECKSUM
PER LA MEMORIA DA 12288 A 12458
DA USARE CON IL PROGRAMMA BASIC
LOADER.

2100  DATA  12288, 0, 4, 40, 39, 24, 7, 32, 30, 12464
2101  DATA  12296, 53, 48, 60, 48, 65, 48, 16, 48, 12682
2102  DATA  12304, 96, 72, 165, 252, 72, 24, 105, 212, 13302
2103  DATA  12312, 133, 252, 160, 0, 173, 134, 2, 145, 13311
2104  DATA  12320, 251, 104, 133, 252, 104, 56, 201, 64, 13485
2105  DATA  12328, 144, 10, 201, 96, 144, 3, 233, 32, 13191
2106  DATA  12336, 96, 56, 233, 64, 96, 32, 228, 255, 13396
2107  DATA  12344, 170, 240, 250, 96, 162, 1, 76, 67, 13406
2108  DATA  12352, 48, 162, 2, 72, 32, 201, 255, 104, 13228
2109  DATA  12360, 32, 210, 255, 96, 1, 0, 0, 0, 12954
2110  DATA  12368, 0, 0, 0, 0, 0, 0, 0, 0, 12368
2111  DATA  12376, 0, 0, 0, 0, 0, 0, 0, 0, 12376
2112  DATA  12384, 0, 0, 169, 3, 174, 76, 48, 168, 13022
2113  DATA  12392, 32, 186, 255, 173, 77, 48, 162, 78, 13403
2114  DATA  12400, 160, 48, 32, 189, 255, 173, 82, 53, 13392
2115  DATA  12408, 133, 253, 173, 83, 53, 133, 254, 169, 13659
2116  DATA  12416, 253, 174, 84, 53, 172, 85, 53, 232, 13522
2117  DATA  12424, 208, 1, 200, 32, 216, 255, 96, 169, 13601
2118  DATA  12432, 0, 72, 40, 32, 189, 255, 169, 1, 13190
2119  DATA  12440, 162, 0, 160, 255, 32, 186, 255, 32, 13522
2120  DATA  12448, 192, 255, 32, 7, 50, 169, 1, 32, 13186
2121  DATA  12456, 195, 255, 96, 255, 0, 255, 0, 255, 13767

---------->  FINE APPENDICE E13  <----------
```

Appendice

Programmi BASIC F

F.1 Routine per caricare il programma oggetto per il Monitor Visibile

```
        APPENDICE F1: PROGRAMMA BASIC 'LOADER' PER CARICARE
                      IN MEMORIA IL CODICE OGGETTO DEL
                               MONITOR VISIBILE

              (E' NECESSARIO AGGIUNGERE A QUESTO PROGRAMMA
               UNA DELLE APPENDICI 'E'. VEDI CAPITOLO 13)
```

```
100 REM       LOADER DI K. SKIER
110 REM
120 DIM BYTE(8)
130 READ PRIMA
140 REM
150 READ ULTIMA
160 REM
170 FOR LINE=PRIMA TO ULTIMA
180 GOSUB 300
190 NEXT LINE
200 PRINT "HO CARICATO SENZA ERRORI DALLA LINEA ";PRIMA;
" ALLA LINEA "; ULTIMA
210 END
220 REM
230 REM
300 READ A
310 SUM=A
320 FOR J=1 TO 8
330 READ BYTE(J)
340 SUM =SUM+BYTE(J)
350 NEXT J
370 READ CHECK
```

```
380 IF SUM <> CHECK THEN 500
381 REM
390 FOR J=1 TO 8
391 REM
400 POKE A+J-1,BYTE(J)
410 NEXT J
420 RETURN
421 REM
490 REM
500 PRINT "CHECKSUM ERRATA ALLA LINEA ";LINE
510 PRINT "INDIRIZZO SPECIFICATO= ";A
520 END
530 REM
540 REM
550 DATA xxx
560 REM
570 REM
580 REM
590 DATA yyy
600 REM
620 REM
```

F.2 Routine per salvare un programma in codice macchina su nastro o disco

```
            APPENDICE F2: PROGRAMMA BASIC PER MEMORIZZARE UN PROGRAMMA
                          IN CODICE MACCHINA SU DISCO O NASTRO



                         (RICHIEDE IL MONITOR VISIBILE INTEGRATO)


10  UNITA =12364
20  LUNG  =12365
30  NOME  =12366
40  MLSAV =12386
50  SETADS=13795
60  :
100 PRINT "SALVATAGGIO CODICE MACCHINA"
110 PRINT
120 INPUT "NOME DEL FILE";NAME$
125 IF LEN(NAME$)>19 THEN NAME$=LEFT$(NAME$,19)
130 POKE LUNG,LEN(NAME$)
140 IF LEN(NAME$)=0 THEN 200
150 :
160 FOR J=1 TO LEN(NAME$)
170 : POKE NAME+J-1,ASC(MID$(NAME$,J))
180 NEXT J
190 :
200 PRINT "NASTRO O DISCO?  (N/D)";
210 GET A$:IF LEN (A$)=0 THEN 210
220 IF A$="N" THEN POKE UNITA,1:GOTO 300
230 IF A$="D" THEN POKE UNITA,8:GOTO 300
240 PRINT "TASTO ERRATO":PRINT:GOTO 200
250 :
300 SYS (SETADS):REM SETTA GLI INDIRIZZI DI PARTENZA E DI FINE
310 SYS (MLSAV):REM SALVA IL PROGRAMMA SU DISCO O NASTRO
```

Appendice

Tabelle di conversione esadecimali-decimali

G

Le due tabelle riportano la conversione tra valori decimali ed esadecimali, rispettivamente per i byte bassi (valori decimali da 0 a 255) e per i byte alti (valori decimali da 0 a 65280), corrispondenti a valori esadecimali da 00 a FF.
Nella prima, per valori decimali superiori a 127, viene fornito anche il complemento a due.

Tabella G.1 Decimali da 0 a 255 — Esadecimali da 00 a FF, byte basso – Complemento a 2

Dec.	Esad.	Dec.	Esad.	Dec.	Esad.	C.a2	Dec.	Esad.	C.a 2
0	00	64	40	128	80	−128	192	C0	−64
1	01	65	41	129	81	−127	193	C1	−63
2	02	66	42	130	82	−126	194	C2	−62
3	03	67	43	131	83	−125	195	C3	−61
4	04	68	44	132	84	−124	196	C4	−60
5	05	69	45	133	85	−123	197	C5	−59
6	06	70	46	134	86	−122	198	C6	−58
7	07	71	47	135	87	−121	199	C7	−57
8	08	72	48	136	88	−120	200	C8	−56
9	09	73	49	137	89	−119	201	C9	−55
10	0A	74	4A	138	8A	−118	202	CA	−54
11	0B	75	4B	139	8B	−117	203	CB	−53
12	0C	76	4C	140	8C	−116	204	CC	−52
13	0D	77	4D	141	8D	−115	205	CD	−51
14	0E	78	4E	142	8E	−114	206	CE	−50
15	0F	79	4F	143	8F	−113	207	CF	−49
16	10	80	50	144	90	−112	208	D0	−48
17	11	81	51	145	91	−111	209	D1	−47
18	12	82	52	146	92	−110	210	D2	−46
19	13	83	53	147	93	−109	211	D3	−45
20	14	84	54	148	94	−108	212	D4	−44
21	15	85	55	149	95	−107	213	D5	−43
22	16	86	56	150	96	−106	214	D6	−42
23	17	87	57	151	97	−105	215	D7	−41
24	18	88	58	152	98	−104	216	D8	−40
25	19	89	59	153	99	−103	217	D9	−39
26	1A	90	5A	154	9A	−102	218	DA	−38
27	1B	91	5B	155	9B	−101	219	DB	−37
28	1C	92	5C	156	9C	−100	220	DC	−36
29	1D	93	5D	157	9D	−99	221	DD	−35
30	1E	94	5E	158	9E	−98	222	DE	−34
31	1F	95	5F	159	9F	−97	223	DF	−33
32	20	96	60	160	A0	−96	224	E0	−32
33	21	97	61	161	A1	−95	225	E1	−31
34	22	98	62	162	A2	−94	226	E2	−30
35	23	99	63	163	A3	−93	227	E3	−29
36	24	100	64	164	A4	−92	228	E4	−28
37	25	101	65	165	A5	−91	229	E5	−27
38	26	102	66	166	A6	−90	230	E6	−26
39	27	103	67	167	A7	−89	231	E7	−25
40	28	104	68	168	A8	−88	232	E8	−24
41	29	105	69	169	A9	−87	233	E9	−23
42	2A	106	6A	170	AA	−86	234	EA	−22
43	2B	107	6B	171	AB	−85	235	EB	−21
44	2C	108	6C	172	AC	−84	236	EC	−20
45	2D	109	6D	173	AD	−83	237	ED	−19
46	2E	110	6E	174	AE	−82	238	EE	−18
47	2F	111	6F	175	AF	−81	239	EF	−17
48	30	112	70	176	B0	−80	240	F0	−16
49	31	113	71	177	B1	−79	241	F1	−15
50	32	114	72	178	B2	−78	242	F2	−14
51	33	115	73	179	B3	−77	243	F3	−13
52	34	116	74	180	B4	−76	244	F4	−12
53	35	117	75	181	B5	−75	245	F5	−11
54	36	118	76	182	B6	−74	246	F6	−10
55	37	119	77	183	B7	−73	247	F7	−9
56	38	120	78	184	B8	−72	248	F8	−8
57	39	121	79	185	B9	−71	249	F9	−7
58	3A	122	7A	186	BA	−70	250	FA	−6
59	3B	123	7B	187	BB	−69	251	FB	−5
60	3C	124	7C	188	BC	−68	252	FC	−4
61	3D	125	7D	189	BD	−67	253	FD	−3
62	3E	126	7E	190	BE	−66	254	FE	−2
63	3F	127	7F	191	BF	−65	255	FF	−1

Tabella G.2 Decimali da 0 a 65280 — Esadecimali da 00 a FF, byte alto

Decimali	Esad.	Decimali	Esad.	Decimali	Esad.	Decimali	Esad.
0	00	16 384	40	32 768	80	49 152	C0
256	01	16 640	41	33 024	81	49 408	C1
512	02	16 896	42	33 280	82	49 664	C2
768	03	17 152	43	33 536	83	49 920	C3
1 024	04	17 408	44	33 792	84	50 176	C4
1 280	05	17 664	45	34 048	85	50 432	C5
1 536	06	17 920	46	34 304	86	50 688	C6
1 792	07	18 176	47	34 560	87	50 944	C7
2 048	08	18 432	48	34 816	88	51 200	C8
2 304	09	18 688	49	35 072	89	51 456	C9
2 560	0A	18 944	4A	35 328	8A	51 712	CA
2 816	0B	19 200	4B	35 584	8B	51 968	CB
3 072	0C	19 456	4C	35 840	8C	52 224	CC
3 328	0D	19 712	4D	36 096	8D	52 480	CD
3 584	0E	19 968	4E	36 352	8E	52 736	CE
3 840	0F	20 224	4F	36 608	8F	52 992	CF
4 096	10	20 480	50	36 864	90	53 248	D0
4 352	11	20 736	51	37 120	91	53 504	D1
4 608	12	20 992	52	37 376	92	53 760	D2
4 864	13	21 248	53	37 632	93	54 016	D3
5 120	14	21 504	54	37 888	94	54 272	D4
5 376	15	21 760	55	38 144	95	54 528	D5
5 632	16	22 016	56	38 400	96	54 784	D6
5 888	17	22 272	57	38 656	97	55 040	D7
6 144	18	22 528	58	38 912	98	55 296	D8
6 400	19	22 784	59	39 168	99	55 552	D9
6 656	1A	23 040	5A	39 424	9A	55 808	DA
6 912	1B	23 296	5B	39 680	9B	56 064	DB
7 168	1C	23 552	5C	39 936	9C	56 320	DC
7 424	1D	23 808	5D	40 192	9D	56 576	DD
7 680	1E	24 064	5E	40 448	9E	56 832	DE
7 936	1F	24 320	5F	40 704	9F	57 088	DF
8 192	20	24 576	60	40 960	A0	57 344	E0
8 448	21	24 832	61	41 216	A1	57 600	E1
8 704	22	25 088	62	41 472	A2	57 856	E2
8 960	23	25 344	63	41 728	A3	58 112	E3
9 216	24	25 600	64	41 984	A4	58 368	E4
9 472	25	25 856	65	42 240	A5	58 624	E5
9 728	26	26 112	66	42 496	A6	58 880	E6
9 984	27	26 368	67	42 752	A7	59 136	E7
10 240	28	26 624	68	43 008	A8	59 392	E8
10 496	29	26 880	69	43 264	A9	59 648	E9
10 752	2A	27 136	6A	43 520	AA	59 904	EA
11 008	2B	27 392	6B	43 776	AB	60 160	EB
11 264	2C	27 648	6C	44 032	AC	60 416	EC
11 520	2D	27 904	6D	44 288	AD	60 672	ED
11 776	2E	28 160	6E	44 544	AE	60 928	EE
12 032	2F	28 416	6F	44 800	AF	61 184	EF
12 288	30	28 672	70	45 056	B0	61 440	F0
12 544	31	28 928	71	45 312	B1	61 696	F1
12 800	32	29 184	72	45 568	B2	61 952	F2
13 056	33	29 440	73	45 824	B3	62 208	F3
13 312	34	29 696	74	46 080	B4	62 464	F4
13 568	35	29 952	75	46 336	B5	62 720	F5
13 824	36	30 208	76	46 592	B6	62 976	F6
14 080	37	30 464	77	46 848	B7	63 232	F7
14 336	38	30 720	78	47 104	B8	63 488	F8
14 592	39	30 976	79	47 360	B9	63 744	F9
14 848	3A	31 232	7A	47 616	BA	64 000	FA
15 104	3B	31 488	7B	47 872	BB	64 256	FB
15 360	3C	31 744	7C	48 128	BC	64 512	FC
15 616	3D	32 000	7D	48 384	BD	64 768	FD
15 872	3E	32 256	7E	48 640	BE	65 024	FE
16 128	3F	32 512	7F	48 896	BF	65 280	FF

Indice analitico

La McGraw-Hill pubblica in tutto il mondo centinaia di libri di informatica per lo studio, la professione e il tempo libero. La produzione in lingua italiana comprende:

88 7700 001 5	J. Heilborn e R. Talbott, *Guida al Commodore 64*
88 7700 002 3	C.A. Street, *La gestione delle informazioni con lo ZX Spectrum*
88 7700 003 1	T. Woods, *L'Assembler per lo ZX Spectrum*
88 7700 004 X	R. Jeffries, G. Fisher e B. Sawyer, *Divertirsi giocando con il Commodore 64*
88 7700 005 8	G. Bishop, *Progetti hardware con lo ZX Spectrum*
88 7700 006 6	H. Mullish e D. Kruger, *Il BASIC Applesoft*
88 7700 007 4	N. Williams, *Progettazione di giochi d'avventura con lo ZX Spectrum*
88 7700 008 2	H. Peckham, *Il BASIC e il PC-IBM in pratica*
88 7700 009 0	H. Peckham, *Il BASIC e il Commodore 64 in pratica*
88 7700 010 4	S. Nicholls, *Tecniche avanzate in Assembler con lo ZX Spectrum*
88 7700 011 2	K. Skier, *L'Assembler per il Commodore 64 e il VIC-20*
88 7700 012 0	S. Kamins e M. Waite, *Programmazione umanizzata in Applesoft*
88 7700 013 9	A. Pennell, *Guida allo ZX Microdrive e all'Interface 1*
88 7700 015 5	P. Cohen, *Grafica e animazione con gli Apple II*
88 7700 016 3	C. Duff, *Guida al Macintosh*
88 7700 017 1	G. Kane, *Il manuale MC68000*
88 7700 018 X	P. Hoffman e T. Nicoloff, *Il manuale MS-DOS*
88 7700 020 1	S. Nicholls, *Grafica avanzata con lo ZX Spectrum*
88 7700 021 X	L.J. Graham e T. Field, *Guida al PC-IBM*
88 7700 022 8	T. Field, *Come usare MacWrite e MacPaint*

In preparazione

88 7700 024 4	H. Peckham, *Il BASIC e gli Apple II in pratica*
88 7700 025 2	C. Morgan e M. Waite, *Il manuale 8086/8088*
88 7700 027 9	G. Mainis, *Il manuale ProDOS*

Un numero sempre più grande di utilizzatori si avvicina alla programmazione in codice macchina che permette di sfruttare totalmente la potenza del microprocessore: il BASIC e gli altri linguaggi ad alto livello infatti sono certamente molto semplici da usare ma, proprio perché devono tradurre ogni singola istruzione dell'utente in un codice comprensibile al microprocessore, non consentono la velocità e la flessibilità richieste dalle applicazioni più raffinate.
La programmazione in codice macchina è però estremamente difficile - se non impossibile - da realizzare senza un appropriato programma assemblatore che permette di scrivere le istruzioni in una maniera relativamente semplice e di tradurle, una volta per tutte, nel codice comprensibile al microprocessore.
L'Assembler per il Commodore 64 e il VIC-20 vuole introdurre alla programmazione in codice macchina in modo inconsueto ma alla portata di tutti: dopo un'introduzione ai principi teorici, infatti, il lettore è invitato a realizzare numerose routine che vengono via via integrate in un completo e funzionale programma assemblatore.
Non è necessaria una preparazione matematica o informatica particolare, perché tutti i problemi vengono affrontati e risolti in maniera estremamente comprensibile e soprattutto perché il lettore costruisce passo dopo passo gli strumenti necessari per risolvere i successivi problemi.
Il valore del libro è quindi duplice, perché al corso di programmazione affianca il listato completo di un potente assemblatore.

Un libro di **BYTE**

McGraw Hill

Lire 35 000
(IVA 2% inclusa)

ISBN 88 7700 011 2